MOVING TO ENTERPRISE SQL

U.S. $8.99 (Canada $9.99)

ColdFusion Developer's Journal

ColdFusionJournal.com

April, 1999 Volume: 1 Issue: 2

From the Editor
Hitting the Ground Running
Chad Sitler page 5

ColdFusion Tag Gallery
A Simple Custom ColdFusion Tag
Jeb Beasley page 16

Slick Tricks
Hidden Frames and JavaScript
Dan Chick page 32

Tips & Techniques
Form to Mail Applications
Jim Esten page 38

CFDJ News
page 42

User Group News
page 15

A Look Inside
ColdFusion 4.0 Developer Enhancements
Charles Arehart page 44

SYS-CON PUBLICATIONS

CFDJ FEATURE page 10

<BF on CF>: Caching in on Performance 6
When used properly, query caching can dramatically improve application performance and response time
Ben Forta

CFDJ Special Feature: Introduction to the Problem 10
Building collaborative workspaces on the Web
Hal Helms

Feature: So You've Outgrown Access *Part 1* 18
Making the move to enterprise-capable SQL Server
Richard Schulze

Feature: Who Says You Can't Take It with You? 26
Keeping your e-mail accessible when you're away from your personal computer
Rich Rosen

CFDJ Feature: What's In It for Me? 34
Taking advantage of new features in ColdFusion 4.0
Michael J. Murdy

Product Review: NetObjects Fusion 4.0 48
Designing great-looking Web pages couldn't be easier with this invaluable tool
Shawn Cowman

IMHO: New Business Models on the Web 50
How to get to the next level from here
Jeremy Allaire

Able Solutions

www.ablecommerce.com

interland

www.interland.net

Eprise

www.eprise.com

ColdFusion Developer's Journal

EDITORIAL ADVISORY BOARD

Geoff Caras, Jim Esten, Ben Forta,
Steve Nelson, Richard Schulze

Editor-in-Chief: Chad Sitler

Art Director: Jim Morgan
Executive Editor: M'lou Pinkham
Managing Editor: Hollis K. Osher
Production Editor: Brian Christensen
Product Review Editor: Tom Taulli
Tips & Techniques Editor: Matt Newberry

WRITERS IN THIS ISSUE

Jeremy Allaire, Charles Arehart, Jeb Beasley,
Dan Chick, Shawn Cowman, Jim Esten,
Ben Forta, Hal Helms, Michael J. Murdy,
Rich Rosen, Richard Schulze

SUBSCRIPTIONS

Subscribe@SYS-CON.com

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department.

Subscription Hotline: 800 513-7111
Cover Price: $8.99/issue
Domestic: $49.99/yr. (6 issues)
Canada/Mexico: $69.99/yr.
Overseas: $99.99/yr
Back Issues: $12 each

Publisher, President and CEO: Fuat A.Kircaali
Vice President, Production: Jim Morgan
Vice President, Marketing: Carmen Gonzalez
Accounting Manager: Ignacio Arellano
Circulation Manager: Mary Ann McBride
Advertising Account Manager: Robyn Forma
Advertising Assistant: Megan Ring
Graphic Designers: Robin Groves
Alex Botero
WebMaster: Robert Diamond
Customer Service: Ann Marie Milillo
Sian O'Gorman
Paula Horowitz
Online Customer Service: Mitchell Low

EDITORIAL OFFICES

SYS-CON Publications, Inc.
39 E. Central Ave., Pearl River, NY 10965
Telephone: 914 735-7300 Fax: 914 735-6547
Subscribe@SYS-CON.com

COLDFUSION DEVELOPER'S JOURNAL (ISSN # 1523-9101)
is published bimonthly (6 times a year)
for $49.99 by SYS-CON Publications, Inc.,
39 E. Central Ave., Pearl River, NY 10965-2306.
Application to mail at Periodicals Postage rates is pending
at Pearl River, NY 10965 *and additional mailing offices.*
POSTMASTER: Send address changes to:
COLDFUSION DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.,
39 E. Central Ave., Pearl River, NY 10965-2306.

© COPYRIGHT

Copyright © 1999 by SYS-CON Publications, Inc. All rights reserved.
No part of this publication may be reproduced or transmitted in any form
or by any means, electronic or mechanical, including photocopy or any
information storage and retrieval system, without written permission.
For promotional reprints, contact reprint coordinator.
SYS-CON Publications, Inc., reserves the right to revise, republish and
authorize its readers to use the articles submitted for publication.

WORLDWIDE DISTRIBUTION by
Curtis Circulation Company
739 River Road, New Milford NJ 07646-3048 Phone: 201 634-7400

DISTRIBUTED in USA by
International Periodical Distributors
674 Via De La Valle, Suite 204, Solana Beach, CA 92075 619 481-5928

All brand and product names used on these pages are trade names,
service marks or trademarks of their respective companies.

SYS-CON PUBLICATIONS

Hitting the Ground *Running*

Well, here's the second issue of ***CFDJ***. We're still the new kids on the block, but now we have one under our belt. The time has come for us to take what we have and make it better. I sincerely hope we've done that and will continue to do so in the future. I'm really enjoying getting all types of article proposals from you folks. I'm also pleased to see the wide array of topics being associated with ColdFusion. CF is touted as a robust e-commerce tool, and this is the truth, but as we're seeing within these pages, there's much more than that to CF. Ben Forta's article on fine-tuning your database cache is a good example. This month we've also included an update on some of the newest features in Allaire's CF beta, 4.0.1. To service our readers on a wider scale, we've added an updated list of CF user groups throughout the world. Be sure to find the CFUG nearest you. You can never learn too much.

A Different Breed

I'm sure you've noticed that ColdFusion developers are a different breed. It kind of reminds me of the high school football team back home in Pennsylvania. Everywhere you go around town you see people wearing T-shirts, sweatshirts and sweatpants with the letters WETSU. These letters stand for We Eat This Stuff Up (heck, they even put it on their cars!). In this town, you see, the order of God, family and country doesn't apply. It's football, then God and country. Football is their religion.

That's a lot like what I've been seeing in many of the CF developers I've been in touch with recently. We eat this stuff for breakfast, lunch and dinner, breathe it as our first breath in the morning and as our last breath at night. We are a committed group, dedicated to making the Web a better place. But enough of the patriot speech. I hope you get my point: I admire that dedication.

Because of this dedication, I want you, our readers, to make sure that we put out a quality publication, issue by issue. Contact me with any comments, suggestions or wisecracks – anything you may have for us. You can also contact our publisher directly at fuat@sys-con.com. He'd like to hear your comments too. And, of course, we're always looking for writers. As I stated in the first issue, we're gearing these articles not only to experienced CF developers, but to the newbies as well. Please don't be intimidated by what's on these pages. If you have an idea, present it. We'd love to hear it.

For the next issue we're working on a few new ideas. One is a page (maybe two) dedicated to snippets of code you send us – a "Tips and Tricks" page, if you will. That's just one possibility. Again, contact me with your suggestions.

Recent Updates

I like to think of my brain as a series of code snippets. I use a certain part to accomplish one task, then move on to the next batch of information to process. (This is my way of indicating that I'm switching gears to a new topic....)

The past few months have been very interesting in the CF universe. I'm sure all of us are keeping an eye on one of the newest IPOs on the market, Allaire. I'm happy to see their steady growth and wish them the absolute best.

Also, we've received the latest copy of our bibles. Ben Forta's *Advanced ColdFusion 4.0 Application Development* recently hit the market, as well as *The ColdFusion 4.0 Web Application Construction Kit.* I'm told by a reliable source that Ben's books were the best-selling Macmillan books on Amazon.com in early February. Congratulations, Ben!

And not to toot our own horn (but I will anyway), ***CFDJ*** appeared on the scene in February to rave reviews. Naturally, we're pleased with the response, and again I ask that you keep the lines of communication open and contact us with any comments or suggestions.

These events can only mean continued success for ColdFusion and all that surrounds it. Personally, I can't wait to see where all of this takes us. CFDJ

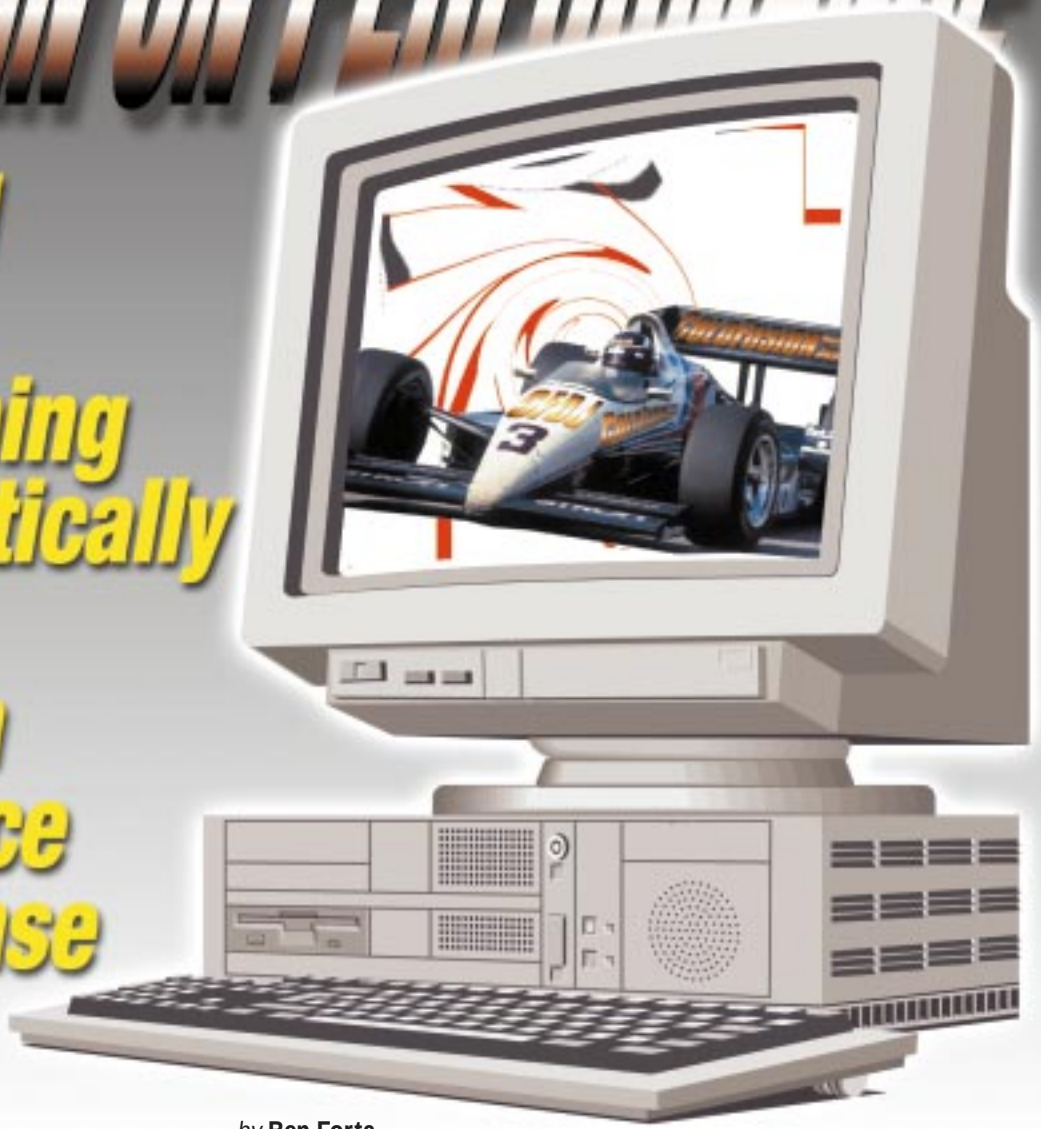
About the Author

Chad Sitler is editor-in-chief of **ColdFusion Developer's Journal**. *Chad can be reached at chad@sys-con.com.*

chad@sys-con.com

CACHING IN ON PERFORMANCE

When used properly, query caching can dramatically improve application performance and response time



by **Ben Forta**

There's nothing that can kill your application's performance as quickly as database access. This is a shame, considering that almost every ColdFusion application you'll ever write will incorporate some sort of database integration.

It thus follows that an important part of optimizing an application's performance is reducing its database activity. And no, this doesn't necessarily mean stripping out database access. The trick is to reduce the amount of database activity that your application generates. This is where caching comes in.

Note: *In this column we'll be looking at database caching only. ColdFusion also features page and p-code caching, but those subjects require columns unto themselves.*

Understanding Caching

Caching is not a new concept, nor is it unique to ColdFusion. The basic principle behind caching is that data that resides in memory can be accessed very quickly – orders-of-magnitude times faster than reading the same data from disk. The hard drive – any hard drive – is still one of the slowest components in a computer. Memory access, on the other hand, is one of the fastest operations a computer can perform.

Caching involves keeping a copy of recent data in memory so that subsequent requests for that data may be fulfilled by accessing the memory-resident copy rather than the original data on the disk.

Many programs feature caching. Most database servers support caching to improve database access time; Internet browsers cache recently used graphics to improve page download time; even operating

systems can cache file system access (does anyone remember the DOS FASTDRIV utility?).

ColdFusion Database Caching

So what does all this have to do with ColdFusion? Well, aside from being able to take advantage of the operating system and database server caching (as any other application would do), ColdFusion features application-level caching that you can use within your own development efforts.

Where would you use caching within your applications? Here are some examples:

- Almost every form that prompts for an address displays a list of states. Those states should never be hard-coded (even though there's no fifty-first state scheduled to join the U.S. at this time); instead, state lists should be populated by a query against a states table. But as that states list doesn't change often (it's 40 years since Hawaii came on board), reading it from the database every time it's needed is a waste of database resources. The states list is thus a primary candidate for caching.
- Employee lists are another good example. While it's true that employee lists can change frequently, it's doubtful that they change so often that they have to be read from the database each time they're needed (if they do, do yourself a favor: find a new employer, and quickly). Caching employee lists for a few hours will reduce database activity, and the only penalty is that personnel changes won't be immediately reflected in your lists.

Even though frequently retrieved data is likely cached by the database server itself, retrieving the data again is obviously more resource intensive than not requesting it at all. Furthermore, as ColdFusion usually isn't running on the same box as the database server, eliminating unnecessary database requests can also reduce network traffic between the two machines, which in turn further eliminates potential performance bottlenecks.

This is where ColdFusion-based data caching comes in. ColdFusion supports two forms of caching: variable-based caching and query-based caching. We'll take a look at both.

Variable-Based Caching

Variable-based caching is a simple concept, and has been supported by ColdFusion since version 3. Most ColdFusion variables persist while a page is being processed and are automatically destroyed as soon as page processing is complete. But ColdFusion features several special variable types that are designed to persist even after a page has finished processing. Table 1 lists some of them.

Variable Type	Description
CLIENT	Client variables are unique to a specific user. They persist until they time out, may be stored in the system registry or in an external data source, and may be shared across hosts in a cluster.
SESSION	Session variables are unique to a specific user. They persist until they time out, are stored in system memory, and cannot be shared between hosts.
APPLICATION	Application variables are unique to a specific application and are shared by all application users. They persist until they time out, are stored in system memory, and cannot be shared between hosts.
SERVER	Server variables are shared by all code running on a ColdFusion server. They never time out, are stored in system memory, and cannot be shared between hosts.

Table 1: ColdFusion persistent variable types

Usually, persistent variables are used to store simple data. For example, if you wanted to count how many requests were made to a specific page since ColdFusion was started, you could use the following code:

```
<CFIF NOT IsDefined("SERVER.pagecount")>
 <CFSET SERVER.pagecount=0>
</CFIF>
<CFSET SERVER.pagecount=SERVER.pagecount+1>
```

The first three lines of this code block ensure that the SERVER variable exists. The inner <CFSET> statement will be processed only once, the very first time the code is executed. After that, the IsDefined() test will always return FALSE because SERVER variables persist until ColdFusion is restarted. The final line of code increments the variable by 1, and will be excepted every time the page is requested.

So what does this have to do with query caching? Well, it's simple, really. ColdFusion allows you to save query results to persistent variables. Look at the following code example:

```
<CFIF NOT IsDefined("APPLICATION.states")>
 <CFQUERY NAME="APPLICATION.states" DATA-
SOURCE="datasource">
   SELECT * FROM states ORDER BY name
 </CFQUERY>
</CFIF>
```

This code checks to see whether an APPLICATION variable named "states" exists. If it doesn't, a <CFQUERY> is executed, and the query is saved to APPLICATION.states. If this code were processed again before the variable timed out, the query wouldn't be executed because it already existed. Once the variable timed out (at the interval specified in the ColdFusion Administrator or in the <CFAPPLICATION> tag), the query would be processed again and the variable would be restored.

The only catch here is that any references to the query must include the specifier APPLICATION. To use the results in a <CFOUTPUT> you'd have to do this:

```
<CFOUTPUT QUERY="APPLICATION.states">
```

The choice of which variable type to use is yours, although you'll probably find that SERVER and APPLICATION variables are the ones you want for most applications.

Because of how variable-based query caching works, the best place to execute (and cache) your queries is in the APPLICATION.CFM file. This way, the first time the application is used all the queries are cached ready for use. On subsequent page requests the cached copies will be used automatically.

Query-Based Caching

Query-based caching is a little different, and is new to ColdFusion 4.0. Unlike variable-based caching, query-based caching occurs right within the <CFQUERY> tag. It is supported by two new <CFQUERY> attributes, as listed in Table 2.

To demonstrate this, let's take a look at the same states list example:

```
<CFQUERY NAME="states" DATASOURCE="data-
source"CACHEDWITHIN=CreateTimeS-
pan(0,0,30,0)>
 SELECT * FROM states ORDER BY name
</CFQUERY>
```

Attribute	Description
CACHEDAFTER	Absolute time-based caching; specifies an actual time after which cached data is cached
CACHEDWITHIN	Relative time-based caching; specifies a maximum amount of time a cache should be used before it is refreshed

Table 2: <CFQUERY> caching attributes

Variable-Based Caching	Query-Based Caching
◗ Always available, cannot be turned off by the Administrator ◗ No limit to the number of queries that may be cached ◗ Data will be cached if requested ◗ Cached queries may be global or user-specific ◗ Cached queries may or may not time out, depending on the variable type used ◗ Depending on the variable type used, cache must be refreshed manually ◗ Code using query results must be aware of caching (and must specify the variable type) ◗ Not well suited for dynamic SQL ◗ Not safe to use for queries with user-specified login information	◗ Can be disabled by the Administrator ◗ Maximum of 100 queries may be cached ◗ Data cached only if the maximum number of cached queries hasn't been reached ◗ Cached queries are global ◗ Cached queries always have a timeout interval ◗ Cache is refreshed automatically as needed ◗ Code using query results is unaware of caching (no code changes needed) ◗ Perfectly suited for dynamic SQL ◗ Safe to use for queries with user-specified login information

Table 3: Variable-based vs query-based caching

In this code example the CACHEDWITHIN attribute is specified using the CreateTimeSpan() function; here an interval of 30 minutes is specified. When the code is executed, ColdFusion will cache the results if it can do so (if the maximum number of allowed cached queries hasn't been reached). No indication about whether or not the results were cached is given, and unlike variable-based caching you need do nothing special to use the query. The following code will work whether or not the data has been cached:

```
<CFOUTPUT QUERY="states">
```

Upon subsequent requests to the page, you may determine whether cached data was used by looking at the debug output at the bottom of the page (if debugging is turned on). Instead of seeing output that looks like this:

```
states (Records=50, Time=40ms)
```

you'd see output like this:

```
states (Records=50, Time=Cached Query)
```

The cache will be used until the specified interval is reached, at which time the data will be reread from the database. Once it has been reread, it will once again be cached if ColdFusion can do so.

It's important to note that unlike variable-based caching, query caching is well suited for dynamic SQL. When the ColdFusion caching engine processes the SQL, it looks at any dynamic statements and even user login information and determines whether the query is the same as one already cached. This means that a query that is built using conditional logic (perhaps <CFIF> statements appending FORM fields) can be cached safely and properly.

Variable-Based Caching vs Query-Based Caching

So which caching mechanism is right for you? Well, the answer is both – it really depends on what you're trying to do. Table 3 lists important points about each cache type.

To help you determine which option will work best for you, consider the following:

- The states list doesn't need to time out, and should be shared by all applications and users (it's highly doubtful that you'd display a different list of states for different users or applications). As such, it's probably best cached as a SERVER or APPLICATION variable.
- "Next n of n style" interfaces are used to allow users to browse through subsets of query results one screen at a time. The way these interfaces are designed usually forces ColdFusion to reread the entire result set each time a subset is needed. As these interfaces are usually driven by user search criteria and need to persist for a limited time (while the user views search results), they are perfect candidates for query-based caching.
- User-specific queries (those containing user-specific information) should be cached using SESSION or CLIENT variables.
- Highly dynamic queries usually benefit least from query caching; not all queries should be cached.
- If you want to ensure that a query has been cached, don't use query-based caching.
- Any data that must always be 100% current should not be cached.

The Fine Print

I have one warning to give you before you run off to cache every query in your application: cached queries can chew up lots of precious memory, and you can't control how much they'll chew up.

ColdFusion lets you specify a maximum number of queries to be cached using query caching, but not a maximum size (in theory, you could cache a hundred queries each of thousands of megabytes of data – I say in theory because in practice that would kill your server before you finished caching them all). For this reason, query caching can be disabled altogether in the ColdFusion Administrator.

Variable-based caching can't really be turned off. While the use of specific data types (e.g., CLIENT) may be prevented, others (e.g., SERVER) cannot. Nor is there a way to restrict how many variables may be created (or the size of their contents).

The bottom line is that while caching can improve performance, abusing caching can do exactly the opposite.

Conclusion

In my experience, 99 out of 100 ColdFusion performance problems are directly related to database access. While not a substitute for good relational database design, appropriate database hardware and efficient SQL, query caching can dramatically improve application performance and response time when used properly.

Considering that almost every ColdFusion application you'll ever write will incorporate some sort of database integration, that's a very good thing indeed. CFDJ

About the Author

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit *and the new* Advanced ColdFusion 4.0 Application Development *(both published by Macmillan).*

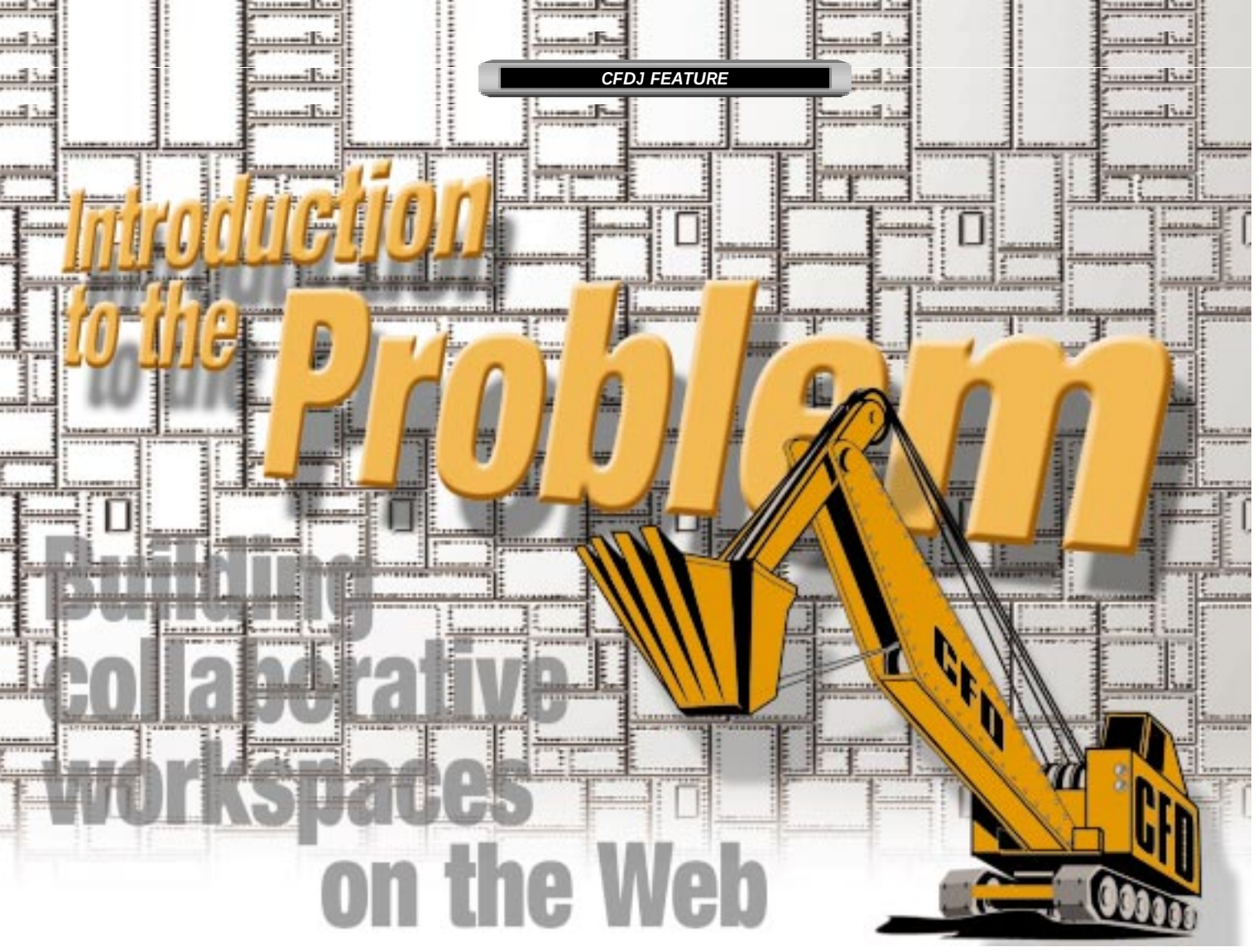
ben@forta.com

Live Software

www.cfanywhere.com

Introduction to the Problem

Building collaborative workspaces on the Web



by **Hal Helms**

Less than a year ago many corporations' vision of the Internet was confined to marketing and advertising. Today, with the dramatic increase of intranets and extranets, developers are being asked to extend the Web into a virtual, collaborative workspace. Many companies want to set up areas of their network as a secure "members only" area. The uses for such an area are extensive and include:

- Virtual collaboration among geographically separate workers
- Customer support areas
- Downloadable evaluation copies, patches and updates

As a ColdFusion developer you can respond to this need quickly. You can partition access to files and set user permissions so different users have distinct views of the files available to them. You can provide an entire "Webplication" that meets the needs of your users and enhances your portfolio. In this article, we'll take a step-by-step approach to setting up the database structure, creating the Web forms and writing the ColdFusion code. If you'd like a copy of the files, contact me at hal.helms@utaweb.com.

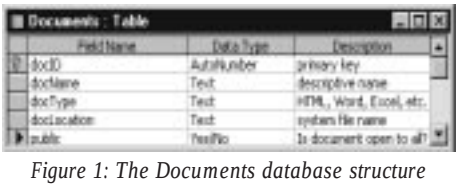


Figure 1: The Documents database structure

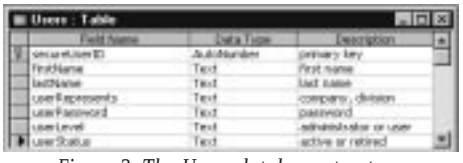


Figure 2: The Users database structure

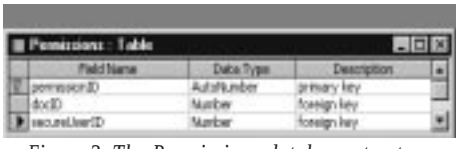


Figure 3: The Permissions database structure

Designing the Database

Few of us get to choose the database we'll work with. Your choice of a database will probably be determined by the standard in place at your client. For this article, we'll illustrate the concepts with Microsoft Access.

We'll create a new database that has three related tables in it: (1) the Users table keeps track of user information, (2) the Files table does the same for files and documents, and (3) the Permissions table decides which users get access to which files. Every record in each file needs a unique identifier or primary key. Microsoft provides an auto-number field in Access that works nicely for this purpose. In addition, add whatever fields each table needs to adequately define both files and users. You can see the database structure I used in Figures 1, 2 and 3.

The database structure is quite simple and merits only a couple of comments. The Users table (see Figure 2) has a field called userLevel. We'll use this to identify which users should be given administrative privileges. In the case of the Permissions table (see Figure 3), the use of a separate primary

key isn't strictly necessary; the combination of a single user with a single file uniquely identifies each row in the table. Adding an auto-number primary key just makes our job a little easier when writing SQL to query these tables.

I've also created a directory structure on the Web server. As we create different forms, we'll use this directory structure to help us with security. Here's a list of the files the application relies on:

- *The root directory contains:*
  1. application.cfm (a.k.a. root_application.cfm) (see Listing 3)
  2. userLogin.cfm (see Listing 1)
  3. doUserLogin.cfm (see Listing 2)
  4. /secure directory
- *The secure directory contains:*
  1. application.cfm (a.k.a. secure_application.cfm) (see Listing 4)
  2. userMenu.cfm (see Listing 6)
  3. /admin directory
- *The Admin directory contains:*
  1. application.cfm (a.k.a. admin_application.cfm) (see Listing 5)
  2. assignDocToUsers.cfm (see Listing 14)
  3. assignUserToDocs.cfm (see Listing 17)
  4. chooseDoc.cfm (see Listing 13)
  5. chooseUser.cfm (see Listing 16)
  6. doAssignDocToUsers.cfm (see Listing 15)
  7. doAssignUserToDocs.cfm (see Listing 18)
  8. docEntry.cfm (see Listing 11)
  9. docList.cfm (see Listing 10)
  10. doDeleteFile.cfm (see Listing 19)
  11. doDocEntry.cfm (see Listing 12)
  12. doUserEntry.cfm (see Listing 9)
  13. userEntry.cfm (see Listing 8)
  14. userList.cfm (see Listing 7)

Creating the Forms

Logging in the User

With the database structure finished, the next step is to create a login form. The code for userLogin.cfm is found in Listing 1.

Files that have forms are processed by "form handlers" specified in the "action" parameter of the <form> tag. In a large project you may create dozens of form files and form handlers. Remembering the purpose of each can be difficult. I use very descriptive names to help in this regard. In files where I'm processing the results of a previous file, I create its name by adding "do" to the filename on which the form itself is located. In this case this yields the form handler "doUserLogin.cfm" (see Listing 2). To understand how they work together, let's look at userLogin.cfm and doUserLogin.cfm.

The user login file begins with several <cfif> statements (see Listing 1, lines 3-29). These will be used to see if the doUserLogin.cfm file is rejecting the login and, if so, for what reason. For the moment, let's skip down to line 34 of Listing 1, where the form begins. Here, users provide their name and password. Note that the user can also click a checkbox if he or she wants to enable the AutoLogin feature. Since users aren't necessarily aware of what's happening "under the hood," you may want to warn them that enabling this feature will allow anyone using their computer the same access they have.

On clicking "OK", this information is passed off to doUserLogin.cfm (see Listing 2). In the form handler, doUserLogin.cfm, a query is executed at lines 2–9 to ensure that the user name and password exist in the database. If so, a session.secureUserID variable is set to the value of secureUserID found in the database (lines 11–13). (In order to create session variables, we must use an application.cfm file [see Listing 3], as discussed below.)

The code at lines 14–17 checks to see if the user has requested AutoLogin. If so, a cookie variable is set with the value of secureUserID in the database. (The next time the user begins the login, the code in userLogin.cfm [see Listing 1] at lines 2–15 will automate the login procedure.)

Later on we'll create pages that are meant to be accessible only to users with an administrative user level. From the previous "validateUser" query, we know the user's userLevel. We can save ourselves some work by creating a session "userLevel" variable now that identifies that level for later use (see Listing 2, lines 18–25).

Once you're done with this, the successful login is sent off to the personalized menu page. Notice the strange way we used to send users on their way? Normally, we would use a <cflocation> tag for this. On this page we encounter a ColdFusion bug: if you have a <cfcookie> on the same page as a <cflocation>, the page will appear to execute flawlessly…but no cookie will be set. Luckily there's a way around this using the <meta> tag. Here's the structure of the tag:

```
<meta http-equiv="refresh"
content="number_of_seconds_to_wait;
url=page_to_send_user">
```

If the login fails, the user is sent back to userLogin.cfm for another try (see Listing 2, lines 26–29). This time, however, a URL variable – badLogin – is set to "yes" and appended to the URL for the user login page.

This is where those <cfif> statements will be used. Lines 18–23 in Listing 1 (userLogin.cfm) look for the presence of this URL variable. Finding it, they inform users that the login failed and ask them to try again.

In addition to providing a form for validating users, our login page needs to perform some other tasks. Earlier, we set a session variable called userLevel to either "User" or "Administrator" if a successful login was made (see Listing 2, lines 18–25). Any attempt to access administrative pages checks the value of this variable. Users without administrative privileges will be transferred back to the login page with the query string "admin=no" appended to the URL. Once there, another of those <cfif>s (see Listing 1, lines 24–29) can handle this event.

Application.cfm

What's to prevent the user from bypassing your carefully constructed user login page by simply typing in the URL of any inner page? To prevent this, we'll use application.cfm files. These files are executed before any other .cfm files within the same directory. This makes them very helpful for holding code blocks that you want to execute prior to your actual page code. For our application, we can place code that verifies that all users have a secureUserID before allowing the file to be processed.

Do you notice a chicken versus egg problem here? The application.cfm file prevents nonauthenticated users from accessing any .cfm pages. But how can users gain access to the login page if they must already be authenticated to get there?

There are many ways to handle this problem. The way this application handles it involves creating a multitiered directory structure. Application.cfm files operate only within their own directory. This allows us to have different application.cfm files, each with their own code for different directories.

We start by placing userLogin.cfm and doUserLogin.cfm in the application root directory. Next we'll create three separate application.cfm files – one for each of our directories. The application.cfm file that goes in our application's root directory (see Listing 3: root_Application.cfm) simply enables session management so we can set the session variable secureUserID.

We'll create another application.cfm file (see Listing 4: secure_Application.cfm) and place it in the secure directory. This file makes sure that anyone trying to access a page in the secure directory has a session.secureUserID. Anyone failing this test is bounced back to the user login screen. Finally, we'll create an application.cfm file to be placed in the admin directory (see Listing 5: admin_application.cfm). This will ensure that only administrators gain access to these files.

Note: The directoryName_ preceding the filename is used to distinguish the different application.cfm files. The files must be renamed "application.cfm" and put in their respective directories before they'll work.

The User Menu

Your clients will probably need varying degrees of security. One popular solution is to provide two broad levels of security. Documents that are marked "public" are available to all authenticated users. Documents marked "private" are restricted – permissions can be set to the level of individual users. Our User Menu (see Listing 6) will handle both of these, as well as provide links to administrative functions for administrators.

Private Files

The SQL for displaying private files (see lines 3–15 of Listing 6) ensures that the user's view sees only those private files they have permission for. The output sets the URL link to the file's actual name while showing the user a friendlier and more descriptive name (see lines 31–42 of Listing 6).

Public Files

In our Documents database table, we created a Boolean field called "public." We can use this field to indicate documents that should be made available to everyone.

The SQL code in lines 16–235 of Listing 6 simply selects all such public documents and the output code (see lines 43–55 of Listing 6) displays them as public files.

I usually place the list of documents in a small frame on the left. The main page is used to display the documents themselves. While that code is omitted here to make the other concepts clearer, you may want to add that to your own pages.

The Administrator's Menu

Finally, lines 56–78 of Listing 6 make access available to the needed administrative functions for those users who are administrators. Notice that in the above code block, the administrative templates are located in the "admin" directory. This scheme allows us to restrict access to non-administrators by putting a small application.cfm file in this directory.

This code (see Listing 5) checks on the value of session.userLevel again. If the user is not an administrator, the <cflocation> tag sends them back to the userLogin.cfm page with the query string "admin=no". We've already written the code to recognize this condition in the userLogin.cfm file (see Listing 1).

User Maintenance

Our application needs to handle basic administrative functions, such as changes to users, documents and the relationships between the two. The first grouping of admin functions centers around the user.

Adding/Editing Users

Notice that a single form (see Listing 7) handles Adding and Editing. Where possible and practical, writing code that handles multiple events will save time and ensure consistency. Here, we begin by listing the users in our database.

The code is written so that clicking on a single user will "drill down" into an edit screen (see Listing 8) for that user. In fact, clicking the "Add New User" link will call the same file: userEntry.cfm. Obviously, we need some way of distinguishing whether we want to edit or add an entry when this form is called.

Again, we'll add a query string to the URL. When we want to use the form to edit an existing user, we'll call it with a query string identifying the user we wish to edit: userEntry.cfm?userID=secureUserID. A single line of code (see Listing 8, line 5) in userEntry.cfm checks to see what mode we're in:

```
5 .<cfif IsDefined("URL.userID")>
```

If the URL variable exists, we must be editing an existing user. The next several lines (see Listing 8) initialize the variables that will hold the values found in the database fields.

The absence of a URL variable means that we're going to be adding a new user. In this case, the variables are initialized with empty strings at lines 23–32. In either case, we can use the same form.

Notice the field called userStatus (see Listing 8, lines 132–136) on this page. One way of removing a user's access to the system is by simply deleting the user record from the Users table. At times you may find it advantageous to keep the information on users in the database while removing the person's access to the system. You can do this by changing the value of the userStatus field from "Active" to "Retired." We check this status during the user login, ensuring that in addition to having a valid username and password, the userStatus is set to "Active."

The next link in this chain is the file doUserEntry.cfm (see Listing 9). This template will handle both updates of an existing record as well as new entries to the database. If we're going to edit a field, we'll need to provide ColdFusion with the primary key of the record to update. So far, we've used session variables for making global variables persistent and URL variables for passing information to other forms. We'd like to make use of the <cfupdate> tag, but this tag expects to get all the information it needs from the form that calls it. We can accomplish this by adding code to userEntry.cfm (see Listing 8, lines 38–44), which includes the primary key of the record in a hidden field.

We'll use the same form handler, doUserEntry.cfm, for both editing and adding users. If we included the primary key as a hidden field, we know that we'll update the database. If it's absent, we'll insert a new record. The code in doUserEntry.cfm (Listing 9) handles this.

Adding/Editing Documents

The code for handling documents is the same as for handling users, differing only in the selection of records from the documents table rather than users. In fact, a good exercise would be to rewrite the code for adding/editing documents and users to use a single form that dynamically adjusts for either users or documents.

A quick glance at docList.cfm (see Listing 10) and docEntry.cfm (see Listing 11) should show the similarities with userList.cfm (see Listing 7) and userEntry.cfm (see Listing 8), respectively. This time, we don't recognize a difference between "Active" and "Retired" documents. However, our delete function only removes the document from our Documents table; it doesn't delete the file from the server and it remains available for you to add it back into the system.

If you still feel the need to make certain documents "inactive" but to retain their information within the system, you can set the documentScope to private and remove all user permissions.

The code that handles docEntry is in the form handler, doDocEntry.cfm (see Listing 12). Note that line 21 of this file sets the des-

Allaire

www.allaire.com

tination – that is, where the uploaded file should physically be stored. You'll need to change this to match the file structure on your server.

Relating Users to Documents

The system administrator will grant permission to individual users to have access to individual documents. One way to do this is to start with the document and then assign users to that document. Or you might switch things about and begin with a user to whom you'll assign documents. Which is best? Ask any administrator and they'll tell you: both. So let's create two different ways of accomplishing the same thing. It's only a little more work and your administrator will love you for it.

Assign a Document to Many Users

We start with taking a single document and assigning many users to that file. First, we must find out what document the administrator wants to set permissions for. The code for this (see Listing 13) provides a list of all private documents. The user's choice is passed by means of a URL variable to the form handler, assignDocToUsers.cfm (see Listing 14).

Now that we know the document the administrator wants to set permissions on, we run a query to find out who currently has access to the document. Lines 3–14 of Listing 14 provide this information.

Next, we want to see all active users. This will determine the universe of users the administrator can select from. Lines 16–27 handle this chore.

The form we create will hold only a few lines of code. This will display all users, indicate the users that were already chosen, and allow the administrator to select or deselect users in this update. This somewhat denser code may not be apparent at first. Let's take it apart:

```
30. <form
31.
action="doAssignDocToUsers.cfm?docID=<cfoutput>#URL.docID#</cfoutput>"
32. method="post">
```

This code snippet creates a form that will call doAssignDocToUsers.cfm (see Listing 15) to take the results of our present file and do the heavy lifting in the database.

```
33. <!--- Outer Loop loops thru ALL users--->
34. <cfLOOP query="getAllUsers">
35. <cfSET selected = "No">
36. <cfSET allID = #secureUserID#>
```

> "Those of us who work to build the Web are on the vanguard of building collaborative workspaces"

We begin to loop through each of the rows returned by the query "getAllUsers"--that is, we're looping through each active user. As we do so, we set a variable called "selected" to "No." We also create a variable called "AllID" that gives us a handle on each ID in the "getAllUsers" query.

```
37. <!---Inner Loop loops through previously assigned users--->
38. <cfLOOP query="getAssignedUsers">
39. <cfSET selectID =
#getAssignedUsers.secureUserID#>
40. <cfIF #allID# is #selectID#>
41. <cfSET selected = "Yes">
42. <cfBREAK>
43. </cfif>
44. </cfloop>
```

Now we begin another – an inner loop – using only those rows that have been previously assigned. We create another handle for each row, called "selectID". Then we check to see if our current outer loop position (AllID) is pointing to a previously selected record (selectID). If so, we set "selected" to Yes and break out of the inner loop.

```
45. <cfIF selected is "Yes">
46. <input type="checkbox" name="selecteUsers" checked value="<cfOUTPUT>#getAllUsers.secureUserID#"</cfOUTPUT>> <cfoutput>
#getAllUsers.lastName#, #getAllUsers.firstName# </cfoutput><br>
47. <cfelse>
48. <input type="checkbox" name="selecteUsers"
value="<cfOUTPUT>#getAllUsers.secureUserID#"
</cfOUTPUT>> <cfoutput> #getAllUsers.lastName#, #getAllUsers.firstName#
</cfoutput><br>
49. </cfif>
50. </cfLOOP>
```

If the row was selected (indicated by the value of the variable "selected"), we can mark the checkbox labeled "selectedUsers" and continue looping through the next row. Users can now select/deselect any boxes they wish to. When they're ready, the submit button sends the form results off to the form handler, doAssignDocToUsers.cfm (see Listing 15).

There, we get the list of users the administrator has selected in the form field "selectedUsers". This is a comma-delimited list that tells us which users should be granted access to the document specified.

There are several ways to handle situations like this one. I prefer a two-step approach that (1) deletes the old permissions and (2) inserts the new ones (see Listing 15, lines 4–23). Wrapping both these in a transaction tells ColdFusion not to execute either of these queries unless both can be done. This provides some protection against system crashes, etc.

Assigning a User to Many Documents

Approaching permissions from the other way around is very similar. The file chooseUser.cfm (see Listing 16) produces the A-list of active users. Drilling down on any of them produces the form assignUserToDocs.cfm (see Listing 17) and its form handler, doAssignUserToDocs.cfm (see Listing 18).

Conclusion

If you've followed the code samples above, you should have a good sense of how to create a useful, though bare-bones, system for handling documents. You may be surprised by just how useful users will find your creation. It's hard to remember that only a very few years ago such universal access to documents was impossible. Those of us who work to build the Web are on the vanguard of building collaborative workspaces. Every step toward this goal will reward users with true empowerment and will put you, the developer, in constant demand. CFDJ

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

About the Author

Hal Helms lives in Atlanta, where, as chief technologist for User Technology Associates, he heads a team of developers working to radically simplify Web development.

hal.helms@utaweb.com

ColdFusion User Groups

Creating a Simple Custom ColdFusion Tag

Writing a custom ColdFusion tag provides better modularity and reusability

by **Jeb Beasley**

This article demonstrates the basic technique for creating a custom ColdFusion tag. You can include custom tags in your templates just as you can any other CFML tag. This mechanism provides better modularity and reusablity. It's also possible to hide your code by encrypting custom tag templates. Before encrypting, be sure to maintain a separate copy of the source. Encrypting can't be undone.

Bear in mind that the server processes all ColdFusion elements. A second version of the tag could wrap JavaScript to process the request. In this case client-side JavaScript would not require a trip to the server.

A Concrete Example

Overview

A custom CFML tag can package logical sections of reusable HTML, CFML or scripting code. Implementation details are hidden in the tag. The tag code resides in a ColdFusion template, but it behaves more like a function or subroutine call. It has a separate scope from the calling template. Variable names don't cross the scope boundary. You pass data into the tag with attribute name-value pairs and you can return data using "caller" attributes. A caller attribute is a tag attribute that carries the name of a variable in the calling template to the tag. I usually instantiate a cfparam to set up a caller variable and pass that variable name into the tag with an attribute. The tag populates that caller variable when it executes (see Listing 1).

Naming and File Location

All template-based custom CFML tags begin with CF_ in either upper- or lowercase. The ending portion of the name must match the base name of the template. For this example, CF_CONCRETE, the associated template must be concrete.cfm. For application-specific tags, place the template in the same directory as the calling template. For tags shared with other applications, store the tag template in the Custom Tags folder to which the ColdFusion Server points. For further organization you can place tags in subfolders beneath the custom tags folder. I've heard it's not a good idea to name the tag the same as the folder.

The Sample CF_Concrete Tag

CF_Concrete is a simple calculator tag. Input parameters are length, width and thickness. Length and width in feet, thickness in inches. It calculates the volume of concrete required in cubic yards, and the weight of the resultant pour in pounds. The tag feeds the results back to variables in the calling scope. Be sure that all attributes are surrounded by double quotes, *not* single quotes. CF does not strip the single quotes and burps.

Begin coding your custom tag with a blank template. All the <html> , <head> and <body> tags are in the calling template. Check for required parameters and validate any data as needed. Process and return. I generally include a debug attribute to toggle display of intermediate results during tag processing.

This example performs a simple calculation and returns data to the caller. A tag can be as simple as a wrapper for printing your name in a consistent format, or it can in itself be a complete application (see Listing 2).

Note the <cfsetting enablecfoutputonly="yes"> tag. When "yes," the tag suppresses display of any text unless it's surrounded by <cfoutput></cfoutput> tag pairs. This greatly reduces excess whitespace output. Be sure to turn it off before returning. Be especially sure to turn it off before any cfabort.

Error Trapping

The tag begins by checking for required parameters. If they all exist, it checks to see that they are numeric. Failure in either case displays an error message on the output page. After outputting the error message, the tag calls <cfabort> with obvious consequences (see Listing 3).

Sample Calling Form

ConcreteTest.cfm exercises the cf_concrete tag. Since the code in the tag is all CF code, the server must process the template. If the tag is implemented in JavaScript, the client browser could perform the calculation without calling the server.

Summary

Custom tags are a powerful extension to the ColdFusion development environment. This extendability greatly improves modularity and reusability of code. It can also provide code hiding through encryption of the template.

This article discusses template-based tags. CFX_ extension tags are DLL based. CFX_ can be extremely flexible, but requires the server administrators to install and register the tags. Some ISPs won't support them. <CFML> template-based tags have no such restriction.

Reference

ColdFusion Users Guide, Allaire Corporation.

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

About the Author

Jeb Beasley has an associate's degree in computer programming/operations from Pace Institute, Reading, Pennsylvania. He is a programmer/analyst who has been developing business and utility applications in C, C++, MFC and VB for 10 years. He also develops Internet applications using ColdFusion with ODBC back-end data sources. Contact him at JebBeasley@BerksNet.com.

jebbeasley@berksnet.com

Allaire

www.allaire.com

So You've

Making the move to enterprise-capable SQL Server

PART 1

by **Richard Schulze**

For many of us, the necessity of moving away from Access is a forced decision. There are a variety of reasons why this may happen. The path from Access to Microsoft SQL Server is a common one. In fact, Microsoft created a special tool – the upsize wizard – just for this reason. But if running the upsize wizard was all you needed to do, this would be a short article.

Although you're reading a magazine dedicated to ColdFusion, you won't see much – if any – information regarding ColdFusion in this article or in the second half scheduled for an upcoming issue. The information here has nothing to do with any Web connectivity software at all. What I'm about to discuss is how to get a desktop back-end database (Access) moved to an enterprise-capable back end (SQL Server).

Several things must be considered prior to this move and to the enhancements afterwards. Not having a basic understanding of how SQL Server stores data or how to optimize the server environment can make your efforts disappointing. Anyone who's spent any time using the upsize wizard and trying to work around error messages knows that the upgrade path isn't always a smooth one.

In this two-part series I'll discuss everything you need to know to make your upgrade a smoother and more beneficial process. I'll cover common errors that the upsize wizard creates and how to work around them. Performance tuning and monitoring of SQL Server, two of the most overlooked and critical pieces of the puzzle, as well as database design will be looked at from a programmer's perspective, using language we can all understand.

outgrown ACCESS

Why Should I Do This and What Can I Expect?

As the former owner of a small ISP with eight servers all running ColdFusion, I can tell you that one of the most common reasons to perform this upgrade is that there's an overload of simultaneous requests. This routinely locks up the server for all the domains running ColdFusion applications on it. Thanks to CF 4.0, this problem is reduced because the service will restart itself when it gets into a bind. However, you still don't want the service stopping and restarting throughout the day.

The growth of the Access database can also cause problems. Depending on their application and use, Access databases can grow very quickly and overrun your allotted hard-drive space, costing the user dollars. As the database increases in size, performance slows along with its growth. Access does have the ability to compact the database, which will reduce the size and bring the performance back in line, but this also has its problems. In order to compact the database you need to download it. Do you down your Web site during this time, write special queries to bring in and/or alter the information in the newly compacted database, with the database still collecting and information online? This also makes it difficult to get that database back onto your Web site without shutting off ColdFusion – not impossible, but difficult.

There are also the obvious reasons for upgrading, e.g., there just may be too many records in various tables, or query performance is slow. I've worked with Access databases with over 250,000 records in some tables. Access 97 can hold this amount (which may amaze some of you), but as the record count goes up, query performance goes down. This is due in part to Access's limited clustering ability, which I'll be covering in detail later.

If you're still reading this article, chances are that one of the above scenarios has probably happened to you. Microsoft SQL Server is the answer to each and every one of these problems. Although you'll probably think you've learned everything there is to know about SQL Server by the time you're done reading this and the follow-up article, that simply won't be true. There just isn't enough room to cover every aspect of this issue in detail. However, to keep it brief, I'll take on each of the above Access limitations and what SQL Server can do to improve those conditions.

One thing to keep in mind is that Access really is a "desktop" database. It isn't built to handle a lot of traffic, such as the kind a busy Web site can develop. Microsoft SQL Server, on the other hand, was built specifically for this. Instead of being restricted by the ODBC driver for Access, your only limitations in most cases are the caliber of the system SQL Server is running on and the bandwidth available to your Web site. As far as a database's growth and subsequent performance problems are concerned, these problems are handled by the fact that SQL Server stores information a lot differently.

Making that statement is kind of like saying that brain surgery is simply opening up someone's head and tinkering around. If you go to your local bookstore and pick up a book on SQL Server, you'll find large sections dedicated to just this information

alone. The bottom line is that your database will start out by eating up a given amount of hard-drive space, and that's all it will ever eat up. In the case of SQL Server 7.0, you can let the devices grow unchecked, but it'll still reflect only the information in it, not a lot of "white space."

The final point to cover is the raw capabilities of Access versus SQL Server. Most Webmasters know that Access begins to die off at 50 simultaneous connections, and that a relatively small database can grow to 20 MG or more in a couple weeks of use. Instead of showing you the relatively small numbers associated with Access, here are the capabilities of SQL Server 6.5:

- *Devices* – 256 devices per SQL Server. Maximum size for each logical device is 32 gigabytes (GB).
- *Databases* – 32,767 databases. The minimum database size is 1 MB. The maximum database size is 1 terabyte (TB).
- *Tables* – Two billion tables per database. The maximum number of bytes per row is 1,962 (excluding text and image columns). The maximum table size is about 1 TB.
- *Columns* – 250 columns per table.
- *Indexes* – One clustered index per table; 249 nonclustered indexes per table. A composite index can have up to 16 columns.
- *Triggers* – Maximum of three triggers per table with one each for INSERT, UPDATE and DELETE. Triggers can have 16 levels of nesting.
- *Stored procedures* – A stored procedure can contain as many as 255 parameters and have up to 16 levels of nesting.
- *User connections* – 32,767
- *Open databases* – 32,767
- *Locks* – Two billion (2,147,483,647)
- *Open objects* – Two billion (2,147,483,647)

I may be wrong, but I think Microsoft SQL Server can handle your needs.

Performance Tuning the Server

Once the workstation is configured to pass volumes of information to the server, the server should be ready to receive it. Keep in mind that SQL Server 6.5 is tuned for a 486 computer with 32 MG of RAM. Most of us have seen the "Minimum Hardware Requirements" before. Usually you don't have to do anything if you have a more capable machine. This isn't the case with SQL Server. If you don't optimize the environment, you'll strangle the server's ability to handle data. Virtually all the changes I'll discuss will be left intact when the server is put into service, and a few are for the upgrade process only. The major difference between normal operation and transferring data from Access really has to do with the size of the log space available and the TEMPDB, or temporary database.

NT Server

There are a couple things to look at in the Windows NT environment before getting to the SQL Server-specific items. In most cases these should have been set already. They'll help ColdFusion and your Web server run more efficiently. You'll find identical information in some of Allaire's white papers. By default, Windows NT is optimized for "foreground" applications. Unfortunately, your Web server, ColdFusion and SQL Server are background applications.

Setting the Application performance couldn't be easier. Simply right-click on your "My Computer" icon and select Properties. Click on the performance tab and ensure that the "boost" slider is set to none (see Figure 1).

Figure 1: The System Properties screen

Also by default, your NT server service isn't set to where it should be for network applications like Web servers, ColdFusion and SQL Server. Sound familiar? Right-click on "Network Neighborhood" and then properties. Select the "Services" tab and double-click on the "Server" service. Ensure that it's set to "Maximize Throughput for Network Applications" (see Figure 2).

SQL Server

The upgrade path I had to follow recently wasn't from a single Access application but from multiple, identical applications representing several separate divisions. This scenario is how the remainder of the article will be represented. Taking the applicable information out for a smaller transition will be a snap.

One of the first things you need to consider is the size of the device to give to the

Figure 2: Customizing the network

database that will ultimately be the real one, i.e., the one that will be going online following the upgrade. There's plenty of information available regarding this determination, but first a bit of background is required, starting with an explanation of what "Devices" are.

Devices can be thought of as of DAT files or sections of your hard drive that are reserved for your database information and transaction logs. Just as the pagefile.sys is always set to a specific size, of which only a small portion is actually used, the same is true of devices. Although these devices can be used to hold data and logs, usually it's best to have separate devices for each. Your database, and the information in it, resides on these devices.

Something to keep in mind is that expanding the size of the device that your database has been created on is impossible in SQL 6.5. Version 7.0 is a bit more forgiving. In fact, 7.0 will allow you either to set a maximum size or to let it grow unchecked so you'll only need to make sure that you're not running out of hard-drive space. Although it's true that you can expand the size of your database by creating additional devices and using them as additional storage, restoring from a catastrophic failure of this configuration is tricky, to put it mildly. Drive space is cheap, so give yourself plenty of room. My rule of thumb is to make your log device 25% the size of your data and make them both 25% larger than you'd expect them to be. Any SQL Server book can help you with this decision.

Once you've made these decisions, the next step is to create the devices and database. I'm going to assume you're using SQL Server 6.5 for the following steps:

1. Start the Enterprise Manager. If you don't see your server listed, you'll need to register it. Your online docs will explain how to do this. The stoplight

Allaire

www.allaire.com

Figure 3: The Microsoft SQL Enterprise Manager

Figure 4: Naming the devices

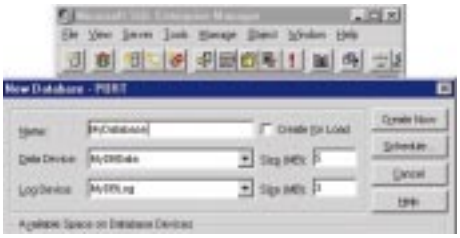


Figure 5: The New Database screen

should also be green, indicating that the SQL Server service is running. If it isn't, start it by right-clicking on the stoplight and choosing start (see Figure 3).

2. Click on the plus sign next to where your server is listed. You'll now see several things, including a folder called "Database Devices."
3. Right-click on "Database Devices" and then "New Device." Name the devices appropriately and create them where and at the size you desire (see Figure 4).
4. Right-click on databases and then "New Database." Use the two devices you just created appropriately (see Figure 5).

You now have a database with absolutely nothing in it. The next issue to be discussed has to do with how SQL Server runs. Remember, it's not set up to run efficiently with any recently built machine and won't take advantage of a powerful machine unless you tell it to. This is where the real performance gains are made. Realistically, SQL Server shouldn't be run on any system with less than a Pentium 200 with 64 MG of RAM. If the server on which you're running SQL Server is also expected to run your Web server and ColdFusion, 256 MG of RAM should be the minimum. Preferably, you'll have SQL Server running altogether separately from your Web server.

The following configuration changes I'll suggest here assume that you'll have at least 256 MG of RAM available in your system. Setting these configuration parameters can be done in two different ways. The first can be done through the query tool and by running various stored procedures. Although this is effective, it's not particularly user-friendly. The suggested method is to use the Enterprise Manager. Simply right-click on your server and click on configure. This will bring up a dialog by which you can set security options, server options, configuration and various other attributes. The configuration tab is the one we're interested in.

In the configuration dialog you can set several default values, but to discuss all of them would be beyond the scope of this article. The configuration options I'm most interested in are free buffers, hash buckets, L.E. threshold maximum, L.E. threshold minimum, Max async I/O, Max lazy write I/O, Max worker threads, memory, opened databases, open objects, procedure cache, RA cache hit limit, RA cache miss limit, RA delay, RA prefetches, RA slots per thread, RA worker threads, show advanced options, tempdb in RAM, priority boost and time slice (see Figure 6).

Figure 6: Server Configurations/Options screen

The parameters I'm about to give are a good starting point for performance-tuning SQL Server. Each individual server will require the Administrator to fine-tune each configuration point to match your needs. Performance monitor can be used to monitor many of these configuration points to aid the Administrator in tuning your server. These options are by no means set in stone but will provide a smoother upgrade process by allowing SQL Server to process data more efficiently. By no means should you consider these the optimum settings for your server/application.

> *"For many of us, the necessity of moving away from Access is a forced decision."*

If you're interested in what these options do, your online docs will provide the necessary information. Simply click in the right-hand column under current and change these options as follows:

- Show advanced options – 1 (This will enable several other options that you may not have seen yet.)
- Free buffers – 1,638
- Hash buckets – 66,250
- L.E. threshold maximum – 200
- L.E. threshold minimum – 20
- Max async I/O – 8
- Max lazy write I/O – 8
- Max worker threads – 255
- Memory – 32,768
- Open databases – 20
- Open objects – 500
- RA cache hit limit – 4
- RA cache miss limit – 3
- Delay – 15
- RA prefetches – 3
- RA slots per thread – 5
- RA worker threads – 5
- Tempdb in RAM – 15
- Time slice – 100
- Priority boost – 1 (Only if SQL Server is to be run on its own machine.)

Getting the Workstation Ready

The first tool you'll need is the upsize wizard for Access 97, assuming you're upgrading from Access 97. You can download it from Microsoft at www.microsoft.com/accessdev/prodinfo/exe/aut97.exe. Once you have it, simply run the executable on the workstation you're upgrading from. After the install program is complete, start Access, open a database and you'll find your new tools listed under tools->addins. The install program assumes you have Access 97 installed already (see Figure 7).

Allaire

www.allaire.com

Figure 7: Access Add-Ins upsizing to SQL Server

As good as the upsize wizard is, there are common problems associated with running it. If you look to the white papers on Microsoft's Web page, most of them get so complicated that they're pretty tough to get much help from. One of the many problems you'll encounter when using the upsize wizard is that it'll time out during large transfers of information. Imagine trying to transfer 250,000 records, and after 15 minutes you get a timeout error and find that none of the records made it. The error will look something like this:

```
ODBC Query Timeout:
Server error 0: Timeout expired
SQL that cause error(s)
UT_CopyData
```

The queries that move data from Access to SQL Server were lengthened well past the default 60 seconds, but if you're moving vast amounts of data, these extended values may still not be enough.

If you start to receive these errors, you have two courses of action. The first is to upsize the structure only and link the tables followed by running an Append query in batches, or by making one easy modification to disable the timeout. I prefer to extend the timeout indefinitely so that all data is transferred at once. The specifics for how to use this tool will be covered in detail, including Append queries, but this portion of the article addresses only the whys and hows of preventing the problem.

First, do a search for WZCS97.MDA, as this holds the bulk of the information on how the upsize wizard works. Once it's located, double-click on it so it opens in Access. You'll be prompted to convert it to Access 97. Whether you do or don't really doesn't matter, as we're not going to be changing the structure.

Now that it's opened in Access, open up the UT_ModGlobals module. This may seem new to some of you because you're not greeted with the normal tables and

Figure 8: Creating a new DataSource

fields you're used to, but are instead faced with pages of code. Perform a search for "Clng(vODBCTimeout)", and you should see the following code:

Figure 9: Creating a new DataSource to SQL Server

DC CFUC

www.teratech.com/cfconf

infoboard

www.infoboard.com

Advertiser Index

Advertiser		Page
Able Solutions www.ablecommerce.com	360-253-4142	2
Allaire Corporation www.allaire.com	888-939-2545	3, 17, 21, 51
Catouzer Inc. www.catouzer.com	604-662-7551	4
Eprise Corporation www.eprise.com	800-274-2814	9
FusionFX www.fusionfx.com	800-780-2422	46
Infoboard www.infoboard.com	800-514-2297	33
Intermedia.net www.intermedia.net	650-424-9935	52
Live Software www.cfanywhere.com	408-996-0300	15
RSW Software www.rswsoftware.com	508-435-8000	27
Schlumberger www.cyberflex.slb.com	800-825-1155	13
SYS-CON Interactive www.sys-con.com	800-513-7111	34,35
The Igneous Group www.igneous.com	877-469-ROCK	28
Virtualscape www.virtualscape.com	212-460-8406	29

```
If Not IsMissing (vODBCTimeout) Then
Qry.ODBCTimeout = Clng(vODBCTimeout)
End If
```

Don't worry about what all of this means; just comment out the first and last lines by putting a single quote in front of them and change the timeout value to zero. You should end up with:

```
'If Not IsMissing (vODBCTimeout) Then
Qry.ODBCTimeout = 0
'End If
```

Your queries should never time out from the wizard. Remember, queries that are caught in some sort of loop will also never time out. If this happens you may be forced to shut off Access or stop SQL Server. Following this modification I have had some timeouts occur but they were for a different reason. Access will now work as long as it has to in order to get the information to SQL Server. When it's all done, however, Access has to wait for an "OK" from SQL Server stating that it's done processing the information. If this period takes too long, you'll get the appropriate error – part of which indicates a timeout. In most cases you can ignore these. Once the information is on SQL Server (even if it's still in the TEMPDB), it will finish up and all your data will have made it. The easiest way to tell is to look at the small progress bar in the lower left-hand corner of your screen when running the upsize wizard. When that progress bar disappears, SQL Server has the information. Just be patient. If it times out you can do queries against the new information in SQL Server and see that all the information has made it.

The last part of the process is to create the DSN connections to the databases you've created on the server. This step can be done while running the upsize wizard, but if a mistake is made it can create a handful of problems. It's usually safer to complete this process ahead of time. To make these connections, open up control panel and then double-click the "ODBC" icon. Select the "System DSN" tab and click "Add" (see Figure 8).

Select the "SQL Server" driver and then click on "Finish." Enter the name you wish to call this connection, an optional description, and then type in the machine name of SQL Server and finally the "Next" button. Leave the authentication method to the NT authentication (default) and click "Next."

The next step is the most important of the entire process. Change the default database to the database you're trying to upgrade to. If you don't, chances are you'll upsize to the master database! (See Figure 9.)

Leave all the other options set on their defaults and click "Next" and "Next" again. Finally, leave the remaining two checkboxes unselected and click "Finish." Perform these steps for each of the databases you'll be working with.

Congratulations! You're now ready to start the upgrade process. Many decisions still need to be made, such as database structure, numbering schemes and the use of indexes, as well as several other issues that simply weren't available with Access.

The second half of this article will cover all the particulars of how to use the wizard, create append queries, optimize the database structure once the data is in place – and more. CFDJ

About the Author

Richard Schulze is the owner and operator of Final Bit Solutions, which specializes in Network/Internet consulting and training. His experience ranges from C and Visual Basic programming to formal network training and experience leading to an MCSE/MCT. He is an instructor at the University of Southern California and head programmer at Motorola for a major intranet application utilizing ColdFusion. You can contact him at rschulze@finalbitsolutions.com.

rschulze@finalbitsolutions.com

RSW Software

www.rswsoftware.com

Who Says You Can't Take It With You?

How to keep your e-mail accessible – even when you're not

by **Rich Rosen**

Have you ever been off on a trip, away from your home or office computer, and wanted (no, needed!) to check your e-mail?

Or have you ever been stuck at a location where firewalls or other network limitations prevented you from accessing your mail or sending mail to others?

E-mail has become a lifeline for many people. For some, it's the main mechanism for staying in touch with others. (Many have been known to check their e-mail more frequently than they check their voicemail or answering machine.) For an increasing number of people, being able to access e-mail is not just a luxury, it's an absolute necessity.

How do you check your e-mail when you don't have access to your own computer? Of course, you need a computer with an Internet connection, but that's just the beginning. In the olden days, before the existence of the Web, it was rather simple. You could telnet to your UNIX host and run an e-mail program like elm or pine.

Nowadays, the process is much more complicated. Most people use POP3-based clients like Eudora, Outlook or Netscape Messenger to retrieve and send e-mail, including attachments. If you don't want to go to the bother of setting yourself up as a "user" of one of these e-mail client programs on someone else's computer, or if you simply can't make use of such programs because a firewall blocks POP3 traffic, another solution is needed.

A Web-based e-mail system is a perfect solution. Users can connect to a Web site that allows them to login, supply their e-mail password (and other relevant information) and view a list of messages for them to read. Unlike the typical POP3 client program that deletes your messages from the server as they're downloaded, this application would leave your messages on the server so that you'd have access to them when you got back to your home base. In addition, you could download the messages you wanted to view selectively, skipping spam and other unwanted e-mail.

This article describes how I used ColdFusion to build a Web-based e-mail system called popART, for "POPmail Access and Retrieval Tool." (I considered calling it "popTART," but feared my lawyer would get a phone call from Battle Creek, Michigan.) The goal was to create a robust, full-featured application for people who are used to more sophisticated e-mail clients like Netscape Messenger, Outlook or Eudora. It may not replace these programs, but it can serve as a viable alternative when they're not available, for whatever reason.

Why Reinvent the Wheel?

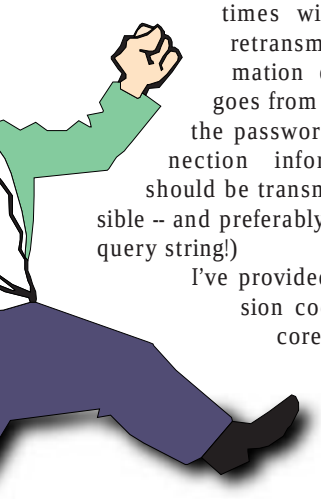
The question arises: Why build a Web-based e-mail system at all? Many portal services already provide a variant of this kind of service. You can use these services to access your e-mail easily over the Web, provided you sign up for an e-mail address on their site. Some of them even let you access mail from your home or office e-mail addresses. But outgoing mail sets your e-mail address to "so-and-so@portal-mail.com" rather than to your original e-mail address. Unless you're looking for anonymity (or extra work for yourself), why get yet another e-mail address?

People who just want access to their home or office e-mail simply need a system that will provide them with that access without requiring them to sign up for a new e-mail address or use proprietary software.

Such a system would be an extremely useful, generalized tool for anyone on the Net to use. It could also be configured as a customized application for a particular group of people who get their e-mail from one particular server, such as the employees of a company or the customers of an ISP.

Why Use ColdFusion?

ColdFusion is a perfect platform choice for building a Web-based e-mail system. Two of ColdFusion's tags, <CFPOP> and <CFMAIL>, are specifically tailored for this very sort of task. But throwing together a ColdFusion module that uses these tags doesn't necessarily make it a viable e-mail system. The key is to maintain "state" between pages so the necessary connection information is known at all times without requiring the retransmission of that information every time the user goes from page to page. (Since the password is part of that connection information, ideally it should be transmitted as little as possible -- and preferably not as part of a URL query string!)



I've provided snippets of ColdFusion code from some of the core modules to illustrate the process of building this application. However, these snippets are simplified versions of the code that I eventually used in my own application, and additional work would be required to create a solid and robust e-mail system based on them.

Requirements

The basic requirements for an application like this are as follows:

1. *A login page*. This page would provide a place for users to enter all the information necessary to connect them to their POP mail server. This would include their POP user name, their password and the name/address of their POP mail server.

   (This page doesn't need to be a ColdFusion module. It can be a simple HTML page with a form on it. The form's ACTION parameter should point to a ColdFusion module that uses the <CFPOP> tag to connect to the server to retrieve mail, using the POST method so that sensitive information isn't included in the URL. The password would be entered here and used as the basis for secure access to e-mail throughout the application, but wouldn't be retained or recorded.)
2. *A message list page*. Once connected, a page should be displayed that lists the "headers" of the messages waiting for you on the server. The header is the part of an e-mail message containing "metadata" information about the message, such as its subject, the address of the person who sent it to you and when it was sent. You'll use the <CFPOP> tag to return the headers of all messages and format them as an HTML table with links to the "bodies" of the individual messages.
3. *A message display page*. Each entry on the message list page should include a link to a page that will display the contents ("body") of that individual message. You can use the <CFPOP> tag here to retrieve the header and body of an individual message.
4. *A means of replying to a message*. By clicking on a link found on the message display page, a user should be able to send a reply to the message. Ideally, similar links should be available to do "reply all" (reply to everyone who got or sent the message) and "forward" (send the message on to someone else). The application can open a window using JavaScript that includes a form with a <TEXTAREA> for entering the response message. This form can link to a page that makes use of the <CFMAIL> tag to actually send the e-mail.
5. *A means of generating a new message*. Similarly, there should be a way to create a new e-mail message from scratch. This function should be usable from everywhere in the application. If you play your cards right, you can reuse some of the functionality you built for replying to messages.

In addition, it would be useful for the application to have a means of viewing or downloading attachments to messages, of deleting messages and of retrieving new mail from the server. This article won't cover these additional features.

Use of Session Variables

A fundamental part of this application is its use of ColdFusion's session variables to maintain "state" as the user jumps from page to page. Under the covers, session variables make use of cookies to identify a pool of values associated with a given application. For this application, the session variables hold the POP server connection information (user name, password, etc.) so that they can persist across pages. This means the application doesn't repeatedly retransmit this information between the ColdFusion server and the browser (though it does transmit the information between the ColdFusion server and the user's POP mail server for each request).

The global ColdFusion module Application.cfm should be used to establish the session variable scope. It can also be used to set other global parameters.

```
<CFAPPLICATION NAME="popart"
   SESSIONMANAGEMENT="Yes"
   SESSIONTIMEOUT=600>
<CFSET SMTPhost = "smtphost.myisp.com">
```

Once session variables are set, they don't need to be included as parameters in links to subsequent pages, either within the query string portion of the URL (for GET) or within the standard input (for POST). The less this information (which includes the password) is transmitted across the wire, the better.

POP3 Protocol and the <CFPOP> Tag

To understand the structure of the application you're building, you need some basic knowledge of the POP3 protocol used to retrieve e-mail on the Internet (see Figure 1).

A very simplified POP3 session goes something like this. By telnetting to port 110, you can "talk" to the POP3 server directly. After the server acknowledges your connection, you can send a line that says "USER username" (substituting your user name, of course). The server will respond by telling you that you need a password, so you send another line that says "PASS password." Once you have passed these authorization tests, you can interrogate the POP3 server about your e-mail by sending additional commands.

The STAT command will tell you how many e-mail messages are waiting. You can use that number as an upper bound for successive applications of the RETR or TOP command, which can retrieve individual messages (header and body) or just the headers from messages. If desired, the DELE command can permanently delete specific messages from the server. (The deletion becomes permanent only when you "commit" by explicitly entering a QUIT command. Messages won't be purged if a session is terminated prior to issuing the QUIT command.)

- USER username = Enter user name
- PASS password = Enter password
- STAT = Obtain number of messages
- RETR n = Get message #n
- TOP n 1 = Get header and first line of message #n
- DELE n = Delete message #n
- QUIT = End POP session and commit mailbox changes

Figure 1: Basic POP3 commands chart

Figure 2: The login page

Figure 3: The main popART window

ColdFusion's <CFPOP> tag can perform sophisticated operations by automatically grouping these commands together. It returns results in a manner similar to <CFQUERY> or <CFDIRECTORY>, so "rows" returned can be iteratively processed via the <CFOUTPUT> tag.

The <CFPOP> tag's required parameters are a USERNAME, a PASSWORD, a SERVER name, an ACTION and a NAME for the <CFPOP> query, which will be used by subsequent <CFOUTPUT> tags. Optional parameters include the MESSAGENUMBER (which could be a range or list of numbers). If the MESSAGENUMBER isn't supplied, it's assumed that the requested operation applies to all messages rather than to just one.

The ACTION parameter can be "getall" (get the entire message including header), "getheaderonly" (get just the header) or "delete" (delete the message from the server). Using ACTION="getheaderonly" without supplying a MESSAGENUMBER returns all of the headers. This format of the <CFPOP> tag is used on the message list page. Alternatively, using ACTION="getall" and supplying a MESSAGENUMBER will return the entire contents (header and body) of a single message. This format is used on the message display page.

The Login Page

The HTML form on the login page will probably need four entry fields: POP user name, POP user password, POP server name or IP address and the user's e-mail address (see Figure 2). In an ideal world, the POP user name and the server name could be derived from the e-mail address. (e.g., "joe@myisp.com" would have a POP user name of "joe" and a POP server name of "myisp.com"). This isn't always the case. Some ISPs give their customers POP user names that are different from the name in their e-mail address, and more often than not the POP server isn't simply "myisp.com," but rather a specific system name like "mailhost.myisp.com" or "popserver.myisp.com".

This page could be customized so that some parameters are hard-coded. This is useful if you're designing this application for employees of a company or customers of an ISP, all of whom use the same POP3 server and all of whom have their e-mail addresses associated with the same domain. You can also create defaults to use if some information isn't entered. For example, the name and domain in the e-mail address can act as the default POP user name and POP server name if those fields aren't supplied.

Message List Page

My version of this application, popART (see Figure 3), makes use of a master page containing three frames. A frame-based architecture can be more efficient, cleaner and more user-friendly because there's less need to use the "Back" button to navigate between the message list and an individual message.

The top frame is a title page containing the popART logo, global buttons used by the application and an ad link generated by an ad server. The middle frame contains the message list page that displays the headers of the messages waiting on the server. The bottom frame starts out blank, but will contain the message display page when individual messages are selected for reading.

If you use a similar design, the form on the login page should connect to this master page. Otherwise it can connect directly to the message list page. First, the parameters passed from the login form are used to set the values of session variables.

```
<CFSET Session.user = #user#>
<CFSET Session.password = #password#>
<CFSET Session.servername = #servername#>
<CFSET Session.address = #address#>
```

Once these values are set, the <CFPOP> tag is used to retrieve the message headers. The output of this tag acts much like a <CFQUERY>, as you'll see below.

```
<CFPOP NAME="email"
       ACTION="getheaderonly"
       PORT=110 TIMEOUT=600
       USERNAME="#Session.user#"
       PASSWORD="#Session.password#"
       SERVER="#Session.servername#">
```

Note that the message bodies (and attachments) aren't retrieved as part of this process, just the headers. This should make the process of retrieving the list faster, but it also affords the user some additional flexibility. If you can see the list of messages before downloading them in their entirety, you can selectively choose *not* to download messages that you don't want to see. You can appreciate this if you've ever used a POP3 client program and waited an eternity for several long spam messages or messages with attachments to download in order to read the one genuinely important message that followed.

First, the parameters passed are used to set the values of session variables.

```
<CFSET Session.user = #user#>
<CFSET Session.password = #password#>
<CFSET Session.servername = #servername#>
<CFSET Session.address = #address#>
```

The message headers are displayed as part of an HTML table. (See Listing 1 for the code snippets from this module.) The first line is for column headings (From, Subject, Received). Subsequent lines are generated by the <CFOUTPUT> tag associated with the query named "email". There are columns for message number, sender (from), subject and date sent. The number displayed in the message number column is a link to the message.cfm module (see Listing 2), passing the message number as a parameter.

Message Display Page

The message.cfm module's primary function is to display the message body (see Figure 4). It begins this process by retrieving the individual message using the <CPFOP> tag.

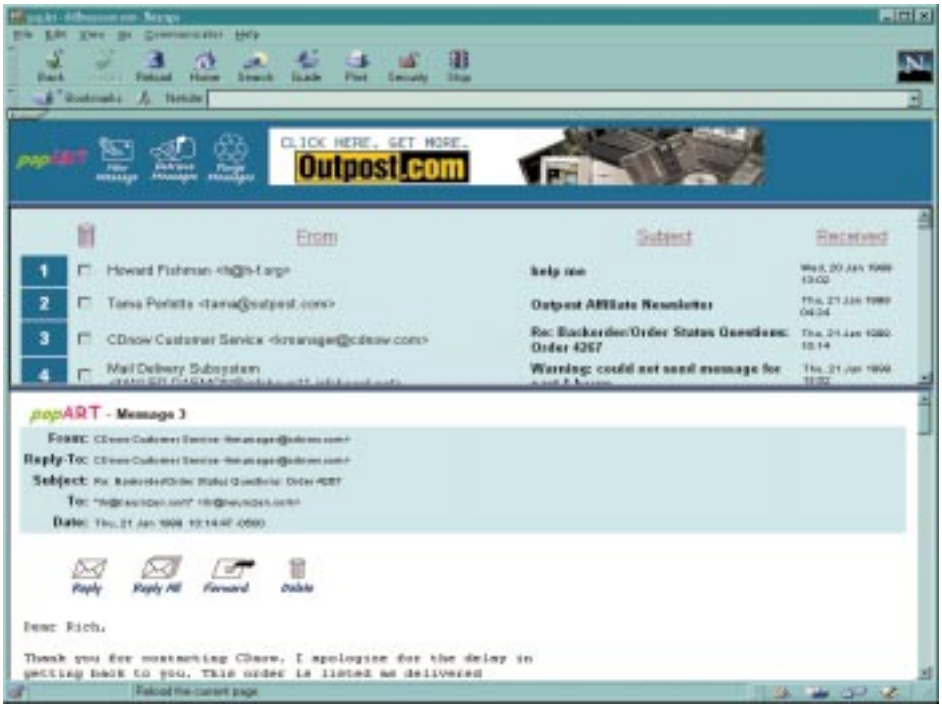


Figure 4: The main popART window with message display

```
<CFPOP NAME="message"ACTION="getall"
       PORT=110 TIMEOUT=600
       USERNAME="#Session.user#"
       PASSWORD="#Session.password#"
       SERVER="#Session.servername#"
       MESSAGENUMBER="#num#">
```

It then redisplays the header information. Following the header, the body of the message is displayed. You could do this using the "<PRE>" tag, but long lines would roll off the edge of the screen. Instead, enclose the body text in a table, explicitly use a monospaced font, and append hard line breaks in the message body with "<BR>" tags.

Reply Function

This would be sufficient if all you wanted to do is display the message. But this module is also supposed to provide a mechanism for replying to the displayed message. To accomplish this task, you must first derive pertinent information from the original message.

You derive the subject and prepend the string "Re:" to it (if it isn't already there). You also derive the reply address, which is the "Reply-to" field (if it exists) or the "From" field.

Finally, you build a copy of the message body, indent each line with "> " and prepend the whole thing with an introductory line. ("On December 31, 1999, so-and-so@somewhere.com wrote:") The JavaScript variable "x" is set to the content of this bodycopy field. The contents of this variable can be retrieved by the subsequent module compose.cfm by referring to "window.opener.x". This is actually far less cumbersome than trying to pass this field as a parameter.

There must also be a link enabling the user to compose a response. In popART, I use an image for this link. The image links to another JavaScript function, com-

Fusion FX

www.fusionfx.com

poseMessage(). I place this image (and other similar images used as links to similar functions) at both the top and bottom of the message display.

As I mentioned earlier, it's our goal to reuse this function for other message composition tasks, such as forwarding, replying to all or generating a new message from scratch. Thus the function may be invoked using different modes to perform these additional tasks. Here, the function simply derives the recipient and subject from the header fields, and uses JavaScript's window.open function to open the compose.cfm module (see Listing 3) in a new, tailored window.

Message Composition Window

This window contains a table with text input lines for each of the header fields and a larger text area for the body of the message (see Figure 5). This area is filled with the contents of window.opener.x, the modified and indented copy of the original message body. Note that the "From" field is hidden, but it can be set up as a regular modifiable text field if so desired.

Sending the E-mail

The send.cfm module (see Listing 4) makes use of the <CFMAIL> tag to send the e-mail message. Note that it makes use of the global variable SMTPhost, which was set in the Application.cfm file earlier. Following the sending of the e-mail, the page tells the user that "Your message was sent" and lists the recipients. (This particular version uses JavaScript's setTimeout function to automatically close this window after eight seconds.)

Drawbacks, Deficiencies and Additional Enhancements

Although the application in its current state is very useful, a number of improvements need to be added to make it a truly robust e-mail application.

1. The "Reply" function should be complemented with a "Reply All" function and a "Forward" function. Most of the core work is already there to support this.
2. The "New Message" function needs to be built. The composeMessage JavaScript function, found on the message display page, has most of the functionality in place to do this. All that's needed is a link that calls this function with a mode parameter of "new." (I place this link in the title frame of my application.)
3. A function for permanently deleting messages would also be nice. Given the way this application interacts with the POP server, the best way to handle this is to "mark" messages for deletion, using a checkbox next to the message number in the message list. A link would be provided to "purge" all the marked messages permanently. This purging would have to be followed by the re-retrieving of all messages from the server to refresh the message list, since deleting messages in the middle of the list would alter the numbers of all subsequent messages.
4. Support for viewing and sending attachments should also be included. The <CFPOP> tag provides support for placing attachments in a specific directory when a message is downloaded. But placing these attachments in a directory accessible through the Web server would make these attachments openly accessible as well. (One commercially available Web e-mail service actually places users' attachments in a publicly accessible directory for *anyone* to see!) For security reasons, the attachments directory shouldn't fall under the Web server's root directory, nor should it be

Listing 1: list.cfm – Message List Page

```
<!--- Get all message headers --->

<CFPOP NAME="message" ACTION="getall"
        PORT=110 TIMEOUT=600
        USERNAME="#Session.user#"
        PASSWORD="#Session.password#"
        SERVER="#Session.servername#"
        MESSAGENUMBER="#num#">
 ...

<!--- Table listing all message headers:
       first line = headings
       (From, Subject, Received)  -->

<TABLE BORDER=0 CELLPADDING=3>
<TR>
 <TD> </TD>
 <TD ALIGN=center><U>From</U></TD>
 <TD ALIGN=center><U>Subject</U></TD>
 <TD ALIGN=center><U>Received</U></TD>
</TR>
<CFOUTPUT QUERY="email">
<TR>
 <TD ALIGN=center VALIGN=center><A
      NAME=msg#email.messagenumber#
      HREF=message.cfm?num=#email.messagenumber#
      TARGET="message_body"
    >#email.messagenumber#</A></TD>
 <TD ALIGN=left
      VALIGN=center>#HTMLEditFormat(email.from)#</TD>
 <TD ALIGN=left
      VALIGN=center>#HTMLEditFormat(email.subject)#</TD>
 <TD ALIGN=left
      VALIGN=center>#email.date#</TD>
</TR>
</CFOUTPUT>
</TABLE>
```

Listing 2: message.cfm – Message Display Page

```
<!--- Get header and body of message #num --->

<CFPOP NAME="message" ACTION="getall"
        PORT=110 TIMEOUT=600
        USERNAME="#Session.user#"
        PASSWORD="#Session.password#"
        SERVER="#Session.servername#"
        MESSAGENUMBER="#num#">
 ...

<SCRIPT>

<!--- Set Javascript variable x,
       referred to later as window.opener.x --->
var x;
function setBodyCopy() {
  x = "#Replace(bodycopy,'""','\""','ALL')#" ;
}

 ...

<!--- composeMessage function --->

function composeMessage(num,mode) {
  recipient = "#URLEncodedFormat(Trim(ReplyAddress))#" ;
```

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

designated as a directory that can be seen via the Web server. In this way, attachments can be viewed or downloaded only through this application, by users authorized to see only their own e-mail attachments.

5. Most of the existing e-mail client programs automatically highlight hyperlinks, such as URLs and e-mail addresses. By clicking on these hyperlinks, you can open up new browser windows or initiate e-mail messages. This is definitely a nice feature to have in an e-mail tool, but it's *very* tricky to figure out where such links begin and end (especially when they end in commas, periods, brackets or other punctuation). ColdFusion's REReplace function is powerful enough to do a lot of this work, but the endings of hyperlinks should be marked carefully. (Also, think about this: Do you want "mailto:" links to open the browser's internal e-mail response function or your application's?)
6. More robust error handling is required. If the POP server doesn't accept the user name and password, the application displays a nasty error message from the ColdFusion server. Also, there's no indication of success or failure when e-mail is sent. Additional error checking and validation are necessary.
7. More robust security is also required. Using ColdFusion 4.0's Encrypt and Decrypt functions might be useful for encrypting passwords. Additional code may be necessary to support "refreshing" of session variables if they've timed out or, if desired, the user can be required to sign in again after a timeout. Using this application over an SSL (Secure Sockets Layer) connection is another way to enhance security.
8. The <CFPOP> tag in ColdFusion 3.0 has some known bugs. One rather serious bug prevents users from reading any message beyond the first one if that first message has an attachment. (This was fixed in ColdFusion 4.0.) It's recommended for this reason that ColdFusion 4.0 should be used as the basis of this application.
9. In addition, the <CFMAIL> tag lacks a lot of the functionality one would want for sending e-mail (blind carbon copy, reply-to header, etc.). There are shareware and commercially available ColdFusion tags that can serve as plug-compatible replacements for <CFMAIL> (and for <CFPOP>), including Christopher Evans's CFX_MAIL and Patrick Steil's CFX_iiPOP3. These are available at Allaire's ColdFusion Developer's Exchange, formerly known as the Tag Gallery (www.allaire.com/developer/gallery.cfm).

Conclusion

I've barely scratched the surface in describing what's involved in building an application like this. The code snippets I've provided, for simplicity's sake, don't include certain finesses in laying out the pages the way they appear in the screen shots. Still, I hope this article serves as a useful introduction to the process of designing and building a Web-based e-mail system in ColdFusion.

If you'd like to take a look at the resulting application in its current state, visit www.neurozen.com/popart. It's still in beta; however, it provides all the main features and a number of the additional enhancements described in this article.

About the Author

Rich Rosen has been on the Net since before there was a Net. He's been with Pencom Web Works (www.pencomwebworks.com) since 1997, building e-commerce, multimedia and Web/database connectivity solutions using NetDynamics, Macromedia Flash, RealAudio and ColdFusion. The popART e-mail system described in this article is one of the many showpieces at his personal Web site (www.neurozen.com).

rlr@neurozen.com

Virtualscape

www.virtualscape.com

Hidden Frames and JavaScript

JavaScript can be a valuable weapon in your ColdFusion arsenal

by **Dan Chick**

"Load. Display. Click. Load. Display. Click."

Sound like a familiar scenario? If you've been doing any kind of Web development you already know that the Web is a request-based environment. Nothing happens unless you're interacting with the Web server and nothing happens unless the Web server sends something back. This is especially true if you're building the data-driven applications that ColdFusion is famous for.

Users are used to filling out a form and hitting a button. They're also used to making a selection from a pull-down and submitting a form to generate the subsequent pages of drill-down pull-downs. Each of these pages carries overhead and potentially several images to download. And maybe it has that annoying little flicker that happens when you reload exactly the same page with one thing changed. Enter the hidden frame and JavaScript…

What You Can Do

Using JavaScript in conjunction with a hidden frame will change the way you do ColdFusion development. Suddenly you have the ability to pump dynamically generated content into existing pull-downs! You can use this frame to run queries and perform database functions without leaving the currently displayed page!

These techniques – and others like them – will result in lightning-fast dynamic pages in your Web application. They'll create the illusion of real-time interaction with the server. You'll essentially be able to create Web applications that feel like Windows applications.

Introduction to JavaScript

If your browsing audience has upgraded their browser in the second half of this decade, then they'll have a JavaScript-compatible browser. Some aspects of the language differ between the Microsoft and Netscape browsers, but JavaScript 1.1 will allow us to do everything we need.

When we're building JavaScript applications with ColdFusion, the most useful situations to capture are when the value of form fields change, when a form is submitted and when a link is hit. For the purpose of this article I'll concentrate only on the form type SELECT.

The techniques that will be presented here won't work if JavaScript is disabled in the client's browser. If you're building a public Web site, then it's risky to trust that JavaScript is something you can count on with a given browser. This also means that if you're developing for an intranet or a controlled extranet, where you can mandate use of a particular browser technology, then you can develop an application knowing that JavaScript is present.

Planning Your Hidden Frame

Your goal is to try to make your hidden frames as invisible as possible. You can venture into DHTML and make them completely invisible, but in the world of traditional HTML your hidden frame will have to consume at least a little screen real estate.

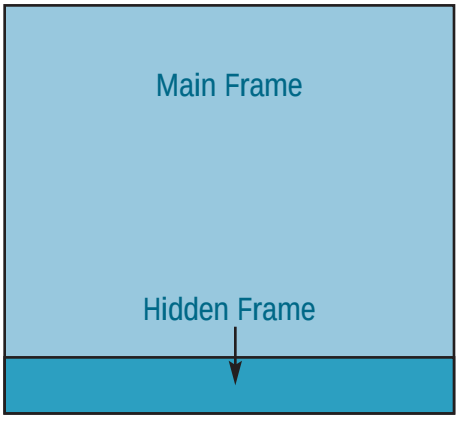


Figure 1: Frameset with a hidden frame at the bottom

Our first hidden frame layout is for a page that's designed to have a nonframes feel. Every time the page changes, the entire page seems to change. You might make your choice of whether to use rows or columns based on your site's background colors and the need to blend your hidden frame in a particular location.

The following code sets up a frameset that looks like Figure 1:

```
<FRAMESET ROWS="*,1"
FRAMEBORDER="0" BORDER=0 BORDER="0">
  <FRAME SRC="main.cfm" NAME="main"
SCROLLING=AUTO>
  <FRAME SRC="blank.html" NAME="hidden"
SCROLLING=NO >
</FRAMESET>
```

If you're creating a frames-based Web site, it's much easier because you can hide your frame at the end of your navigation frame. This will cut back the hidden frame size by about 80%. See Figure 2 for a guideline, then build it like this:

```
<FRAMESET COLS="100,*"
FRAMEBORDER="0" BORDER=0 BORDER="0">
  <FRAMESET ROWS="1,*"
FRAMEBORDER="0" BORDER=0 BORDER="0">
    <FRAME SRC="blank.html" NAME="hidden"
SCROLLING=NO >
    <FRAME SRC="menu.cfm" NAME="menu"
SCROLLING=NO>
  </FRAMESET>
  <FRAME SRC="main.cfm" NAME="main"
SCROLLING=AUTO>
</FRAMESET>
```

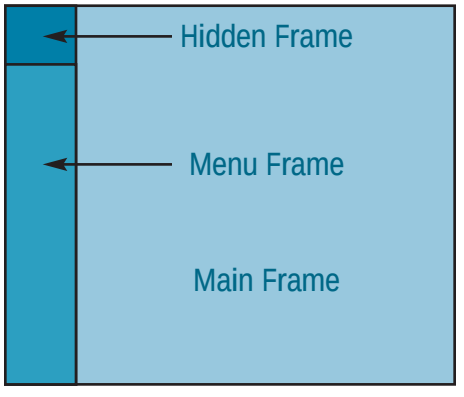


Figure 2: Frameset with a hidden frame in the navigation column

In any case, it's probably wise to keep your hidden frame on the top level of your frameset hierarchy so you don't have trouble referencing it.

To illustrate these concepts in practical terms, let's go straight to two examples (the example code is available for you to download).

Example 1

A user has already logged in and I'm tracking a variable called UserId. In a live application it would be passed in through a variable, but for our example I'll just set it at the top of the file. The page the user is looking at enables him or her to select a client and view all relevant company notes.

I'll be using the "examples" datasource. The information being sought is in the ClientMemo table and the view-tracking records will be stored in the ReadMemos table. For the sake of table simplicity, and perhaps because we're all used to dealing with databases that haven't been normalized, the ClientMemo table has the client name and a unique ID on every line.

ColdFusion is good at populating SELECT boxes so when the page loads it'll contain a list of the clients. What's special is the JavaScript in the onChange event. When the user chooses a client from the pull-down, JavaScript sends a document request to the hidden frame with the UserId variable and the value of the ClientNumber that was selected. Notice how I enclosed my JavaScript function in a CFOUTPUT tag to build the UserId variable right into the location call.

The getmemo.cfm file logs the request, then fetches all the memos for the client. What follows is a CFOUTPUT loop through the query as I incrementally build up a JavaScript variable to pass back to the main frame. When the loop is completed, the TEXTAREA in the main frame is updated with the appropriate value. No visible HTML code has been displayed at all in the hidden frame, only some invisible JavaScript. You'll need to remember to include a BODY tag if your hidden frame is supposed to blend into a certain color.

Listing 1 shows the code that runs in the hidden frame.

So that's it! We've seen ColdFusion create JavaScript statements and we've seen queries going on behind the scenes. If you were paying attention you may have noticed that you can create JavaScript variables with ColdFusion in a single pass or in multiple passes. You can't, however, create ColdFusion variables with JavaScript unless you actually refresh a document somewhere and build the variable into the query string.

Example 2

Think of how easy it would be for users to use an application based on intelligent SELECT boxes. They pick a category in the first box and only the options that are relevant to their first choice show up in the second box. Then they change their minds and pick a new category, and the contents of the second SELECT immediately reflect that. This second example will illustrate how to do that in a JavaScript-enabled browser.

For the purposes of illustration I'm using an overly simple example. The first SELECT will contain the letters A, B and C. Changing this letter in the SELECT will query a table and dynamically populate the second SELECT with the words from the database that start with that letter.

Listing 2 shows the code that runs in the hidden frame.

Looking at the code you can see that when the SELECT box is changed, the getmatches function is called. If a letter is chosen, then control is passed to the hidden frame. There the database is queried to retrieve the data that should populate the results SELECT box. Finally, the query results are looped, adding the data row by row. What sets this example apart from the first one is the way the hidden frame calls a JavaScript function in the main frame to pass the data back to the main frame. The first example executed JavaScript on its own to update the main frame.

You may already be able to see the exciting possibilities this presents. Any query that you want to run can pass data to any JavaScript function you want to have waiting without even requiring the reloading of the page! You'll be able to create drill-down SELECT boxes, validate data on the fly on a per-field basis and more.

Think about it: when people type their last name in an information field, you can instantly populate the rest of the form with their information if they already have a listing in the database. There's a lot of power in adding the hidden frame approach to your ColdFusion toolbelt.

To see several examples of these techniques in action, visit our FusionCart demo store (www.fastcart.com/rockstore) and look in the admin section. Most advanced data-handling aspects use the hidden frame technique. CFJ

About the Author

Dan Chick is the lead developer for FastCart Corp., a Minneapolis-based e-commerce firm. He recently headed the development of FusionCart, a ColdFusion-based shopping system. Dan also runs the Twin Cities CFUG. You can contact him at chickd@fastcart.com.

chickd@fastcart.com

Listing 1

```
<!--- get a list of all the memos for this client --->
     <CFQUERY DATASOURCE="example1" NAME="memoquery">
             SELECT DateEntered, MemoText
             >FROM ClientMemo
             WHERE ClientNum = #ClientNum#
             ORDER BY DateEntered
     </CFQUERY>

     <SCRIPT>
     var WholeMemo = "";

     <!--- loop through the results and build up the return
string --->
     <CFOUTPUT QUERY=memoquery>
     WholeMemo += "Date: #Month(DateEntered)#/#Day(DateEn-
tered)#/#Year(DateEntered)#\n";
     WholeMemo += "#Replace(MemoText, ' " ', ' "" ', "All")#
     WholeMemo += "\n----------------------------\n"
     </CFOUTPUT>

     <!--- this part sets the value in the other document --->
     top.main.document.exampleform.TheMemo.value = WholeMemo;

     </SCRIPT>
```

Listing 2

```
<!--- Query to get the list of matches --->
     <CFQUERY DATASOURCE="example" NAME="allTheMatches">
             SELECT TheWord, WordNum
             >FROM TheWords
             WHERE TheWord LIKE '#firstletter#%'
     </CFQUERY>

     <SCRIPT>
     var DestDoc = top.main.document;
     var DestField = SourceDoc.exampleform2.matches;
```

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

What's In It for Me?

Taking advantage of new features in ColdFusion 4.0

Part 1

by **Michael J. Murdy**

If you're as busy as I am, chances are you seldom have the time to sift through all the support documentation that accompanies your new authoring software – especially if that software is an upgrade to one of your familiar tools. I generally install it and run (we don't need no stinking directions!). In December 1998, I downloaded Allaire's ColdFusion 4.0, the upgrade for one of my favorite and most trusted applications. Fortunately, I actually read the documentation this time.

ColdFusion 4.0 includes many powerful new features that demonstrate Allaire's continuing commitment to this industry-leading technology. In this three-part series I'll share some of the features I've discovered and how I'm putting them to use. Who knows? Maybe you'll decide to start reading your documentation too.

In this issue I'll introduce you to several new tags and functions found in 4.0, and cover some of the ways you may want to implement them. In upcoming issues of ***ColdFusion Developer's Journal*** we'll explore performance tuning and improved security features.

New <Tags> Available in ColdFusion 4.0

The following section covers several of the new ColdFusion tags that you can probably put to use in your code immediately. Additional new tags (not included here) provide advanced functions such as access to registry settings, stored procedures, Verity collections, user authentication, CFScripting and WDDX formatting, which are beyond the scope of this article. For more information on these tags refer to Advanced ColdFusion Development in your documentation or explore the other ColdFusion developer resources available (see sidebar).

<CFLOCK>

CFML Language Reference definition: *The CFLOCK tag single-threads access to the CFML constructs in its body. Single-threaded access implies that the body of the tag can be executed by at most one request at a time. A request executing inside a CFLOCK tag has an "exclusive lock" on the tag. No other requests are allowed to start executing inside the tag while a request has an exclusive lock. ColdFusion issues exclusive locks on a first-come, first-serve basis.*

Syntax

```
<CFLOCK
   NAME="lockname"
   TIMEOUT="timeout in seconds"
   THROWONTIMEOUT="Yes/No">
<!--- CFML to be synchronized --->
</CFLOCK>
```

I'm using the CFLOCK tag to maintain integrity in several application processes. File manipulation routines (read, write, move, delete, etc.) that must be executed by just one user at a time are one appropriate use of this new tag. Wrapping CFLOCK tags around application constructs that perform database modifications (update and insert queries) helps maintain database integrity, especially when used in conjunction with the CFTRANSACTION tag that rolls back any modifications if the entire block is not executed (see Listing 1). In fact, any process in your code that needs to be protected from multiuser access should be wrapped with the CFLOCK tag. This also applies to CFXs (custom CFX tags) that aren't implemented in a thread-safe manner.

The CFLOCK tag will accept the optional THROWONTIMEOUT attribute to configure how your application proceeds if an exclusive lock can't be achieved within the specified timeout period. The default is THROWONTIMEOUT="Yes", which throws an exception. The alternative, THROWONTIMEOUT="No", will pass control to the next line of code following the </CFLOCK> tag.

Hint: Be careful to set the TIMEOUT attribute to the minimum value necessary to avoid clogging your throughput. Otherwise you may create a bottleneck in your application as users stack up while trying to access the code within the CFLOCK block.

"ColdFusion 4.0 includes many powerful new features that demonstrate Allaire's continuing commitment to this industry-leading technology."

<CFTRY>, <CFCATCH> and <CFTHROW>

CFML Language Reference definition: *Used with one or more CFCATCH tags, the CFTRY tag allows developers to catch and process exceptions in ColdFusion pages. Exceptions include any event that disrupts the normal flow of instructions in a ColdFusion page such as failed database operations, missing include files, and developer-specified events. The CFTHROW tag raises a developer-specified exception that can be caught with CFCATCH TYPE="APPLICATION" or CFCATCH TYPE="ANY".*

Syntax

```
<CFTRY>
   ... Add code here
<CFCATCH TYPE="exceptiontype">
   ... Add exception processing code here
</CFCATCH>
   ... Additional CFCATCH blocks go here
</CFTRY>
```

The combination of these exception-handling tags provides ColdFusion programmers with greater control over what happens if the train comes off the track:

By wrapping code in a CFTRY block (which contains at least one CFCATCH tag), you can display special instructions or execute specific commands when a predetermined exception occurs. In fact, you can wrap an entire ColdFusion template with a CFTRY tag, using a CFCATCH block around potential errors.

The TYPE attribute of the CFCATCH tag recognizes the following types of exceptions:

- **Application** – Developer-defined exceptions configured with the CFTHROW tag
- **Database** – Failed SQL statements, ODBC errors or other database failure
- **Template** – General ColdFusion template errors

- **Security** – Exceptions involving security functions
- **Object** – Exceptions in code working with objects
- **MissingInclude** – Exceptions when CFINCLUDE targets are missing
- **Expression** – Exceptions thrown by failed expression evaluations
- **Lock** – Generated when CFLOCK blocks timeout or fail at runtime
- **Any** (default) – All of the above, plus unrecognized exceptions

ColdFusion tests CFCATCH tags in the order they appear on the page, so you'll obviously want to place CFCATCH TYPE="Any" last if you are catching more than one type of exception. Since the TYPE="Any" attribute will capture any exceptions thrown, subsequent CFCATCH blocks will not have the opportunity to catch exceptions.

Hint: The CFTRY block will not handle exceptions thrown by the CFCATCH blocks that it encloses. This means that the code within any CFCATCH block should be relatively fail-safe or the flow of your application will be interrupted and the whole purpose of your CFTRY/CFCATCH construct will be defeated.

The CFTHROW tag is used in conjunction with the CFTRY and CFCATCH tags to generate developer-specified exceptions. Using the CFIF conditional is a good way to implement this technique (see Listing 2).

Caution: Documentation indicates that attempting to handle unexpected exceptions in CFML code can cause unpredictable results and may seriously degrade or crash the ColdFusion Server. Although I haven't experienced problems with this, it warrants consideration as you develop your exception handling routines.

<CFSWITCH>, <CFCASE> and <CFDEFAULTCASE>

CFML Language Reference definition: *Used with CFCASE and CFDEFAULTCASE, the CFSWITCH tag evaluates a passed expression and passes control to the CFCASE tag that matches the expression result. You can optionally code a CFDEFAULTCASE tag, which receives control if there is no matching CFCASE tag value.*

Syntax

```
<CFSWITCH EXPRESSION="expression">

  <CFCASE VALUE="value" DELIMITERS="delim-
iters">
  HTML and CFML tags
  </CFCASE>

  additional <CFCASE></CFCASE> tags

  <CFDEFAULTCASE>
  HTML and CFML tags
  </CFDEFAULTCASE>

</CFSWITCH>
```

The CFSWITCH/CFCASE construct provides better performance and is easier to read than a series of CFIF/CFELSEIF tags. First, define a condition with the CFSWITCH tag. Then add one or more CFCASE tags within the CFSWITCH block, optionally ending with a CFDEFAULTCASE tag. The CFSWITCH tag selects the CFCASE tag that matches its specified condition and executes the code between the CFCASE start and end tags.

This program control structure is especially useful when you need to sift through a large amount of data returned by a database query (see Listing 3).

New Functions( ) Available in ColdFusion 4.0

This section introduces several new functions you may find immediately useful. Once again, there are several new functions we won't be able to address within the scope of this article; most of them pertain to working with structures and new security features. Consult your documentation and other ColdFusion resources for more information (see sidebar).

Netamorphasis

www.netamorphasis.com

ColdFusion Resources

Allaire Alive
http://alive.allaire.com/

Allaire Alive combines the power of a ColdFusion application with streaming media to bring ColdFusion developers information and training from the source - the development team that builds Allaire's ColdFusion products.

ColdFusion Developer Center
www.allaire.com/developer/ReferenceDesk/

The Developer Center provides numerous resources and industry news for CF developers. You can usually get a feel for what others are doing with ColdFusion and pick up useful hints and tricks.

ColdFusion Developer's Journal
www.ColdFusionJournal.com

CFDJ, the first and only CF magazine, also offers a wealth of information online. After perusing ***CFDJ***'s Digital Edition and downloading a storehouse of code from some of the top CF programmers in the industry, you can check out ***CFDJ***'s ColdFusion Forum. Access is free.

ColdFusion Support Conference
http://forums.allaire.com/devconf/main.cfm

This is a great forum for posting questions and receiving prompt and accurate answers. Take advantage of the ColdFusion Forums platform by requesting e-mail notification when postings are added to your thread, and you won't have to check the site every two hours to see if your question's been answered!

CF4 release notes
www.allaire.com/Handlers/index.cfm?ID=6357&Method=Full

Read this article in the knowledge base to familiarize yourself with some of the known issues regarding the implementation of ColdFusion 4.0.

Ben Forta's ColdFusion Page
www.forta.com/cf/

Ben Forta is the author of several outstanding reference books on ColdFusion development and a member of ***CFDJ***'s Advisory Board (as well as a writer for the magazine), and is recognized as an authority on ColdFusion. The site provides links to purchase his books online.

CFMCentral.com
www.cfmcentral.com/

This site features how-tos, tips and tricks, site listings and interaction with other CF developers. Free registration (e-mail address) required to gain access.

- **DirectoryExists(absolute_path):** This function returns YES if the directory specified in the argument does exist and NO if it doesn't.

Several of my applications create user-named directories, which means that my applications must check for the existence of a directory before a new one can be created. In the past, I've had to write and save a marker file in a new directory when it was built and then use the FileExists(absolute_path) function to check for the marker file as a workaround. I now use this function to check for duplicate directory names (see Listing 4). This saves time and resources, making my life easier (always a good thing).

- **Encrypt (string, key):** This function allows encryption of the specified string.
- **Decrypt (encrypted_string, key):** This function decrypts an encrypted string.

These two new tags add encryption capabilities to your set of ColdFusion tools. If you need to exchange sensitive data in your application, these functions allow you to scramble (encrypt) strings that can be sorted out (decrypted) only if the exact key value that was used to encrypt the string is present in the decrypt function (see Listing 5).

Listing 1

```
<HTML>
<HEAD>
   <TITLE>Example CFLOCK</TITLE>
</HEAD>
<BODY>
 <!--- open CFLOCK block --->
<CFLOCK TIMEOUT="5" NAME="Impeachment">

<CFTRANSACTION>

<CFQUERY NAME="Dump" DATASOURCE="cfsnippets">
   UPDATE EMPLOYEES
   SET Email = 'FIRED'
   WHERE Department = 'Slacker'
</CFQUERY>

<CFQUERY NAME="Fetch" DATASOURCE="cfsnippets">
   SELECT * FROM EMPLOYEES
</CFQUERY>

</CFTRANSACTION>

We fired:
<UL>
<CFOUTPUT QUERY="Fetch">

<CFSWITCH EXPRESSION="Department">

   <CFCASE VALUE="Slacker">
   #Fetch.LastName#.<BR>
   </CFCASE>

   <CFDEFAULTCASE>
   Bill Clinton.<BR>
   </CFDEFAULTCASE>

</CFSWITCH>

</CFOUTPUT>
</UL>

</CFLOCK><!--- close try block --->
</BODY>
</HTML>
```

Listing 2

```
<HTML>
<HEAD>
   <TITLE>Example CFTRY/CFCATCH</TITLE>
</HEAD>
<BODY>

<CFTRY> <!--- open try block --->

I see the train is coming...

<!--- condition for exception --->
<CFIF 1 IS 2>
   <!--- throw exception & message --->
   <CFTHROW MESSAGE="We have a problem!">
<CFELSE>
   <H2>All Aboard!</H2>
</CFIF>

<!--- catch exception --->
<CFCATCH TYPE="application">
```

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

I have a third-party security certificate and utilize the SSL (https://) protocol in all of my commerce applications, so I haven't had any pressing need for these functions. I have experimented with them, however, and I'll probably find a use for them in the future. Look for more about security features of ColdFusion 4.0 in the third part of this series.

- **GetTickCount():** Returns a millisecond clock counter that can be used for timing sections of CFML code or any other aspects of page processing.

Although the CF Application Server provides logging triggered by page access times, this particular function gives you an easy way to monitor the performance of specific sections of your code. By setting a variable before and after a block of code and then comparing the values, you can use a CFSWITCH/CFCASE construct to trigger predetermined actions. For example, if the execution of a database query exceeds 3000 milliseconds, you may want to execute code that sends the administrator an e-mail message or perhaps triggers an optimization routine (see Listing 6).

- **REFindNoCase (reg_expression, string [, start] [,returnsubexpressions]):** Returns the position of the first occurrence of a regular expression in a string starting from the specified position. Returns 0 if no occurrences are found. The search is case-insensitive.
- **REReplaceNoCase (string, reg_expression, substring [, scope]):** Returns string with a regular expression being replaced with substring in the specified scope. The search is case-insensitive.

The regular expression functions are powerful tools that provide you with a mechanism for searching and matching text. These functions are used in conjunction with string manipulation functions, usually in file operations. For example, if you needed to write code that locates the second occurrence of the text "Sample Target" in a file, but you're not sure if the actual text is "sample target" or "SAMPLE Target" or even "Sample Target," you could weave a complex series of searches and conditional statements to accomplish this, but regular expressions give you a tool that can achieve this feat in one fell swoop (see Listing 7).

The "No Case" aspect of these two tags brings a new twist to these tools. In previous versions of ColdFusion, if you wanted to use regular expressions to execute a case-insensitive search, you needed to code that in the regular expression string itself, which is a fairly complicated affair. These new tags make this powerful search technique much easier to implement. For more information on Regular Expressions and their use in string manipulation, I suggest reading *Mastering Regular Expressions* by Jeffrey E. F. Friedl.

I'd like to make two points in closing. First, ColdFusion 4.0 contains a wealth of new features and functions that build upon previous releases to define a powerful platform that scales well and deploys rapidly. Since you're reading this journal I'll assume that I'm preaching to the choir. Second, the widespread acceptance of ColdFusion has generated a growing community of CF developers who offer numerous opportunities for you to find quick answers to your programming problems. Use them – and remember to pass your knowledge on to others who seek assistance.

About the Author

Michael J. Murdy is a founding partner and lead developer for FastTrac International, specializing in Internet applications for business. ColdFusion is a central component of their Internet solutions. Contact Mike by e-mail at murdy@FastTrac-INTL.com or visit the FastTrac Web site at www.FastTrac-INTL.com.

murdy@FastTrac-INTL.com

SiteHosting

www.sitehosting.net

New Life for Form to Mail Applications

Extending the form mailer to include archiving inputs to a database

by **Jim Esten**

When choosing the tools for new projects, I must confess that my first inclination usually tends toward a CGI solution. For what has now become several years, I've extolled many of the virtues of CGI. No one is going to argue the sheer numbers. There are just plain more free and low-cost CGI scripts available to do real work than there are ColdFusion templates. The Tag Gallery is one of my favorite haunts, but let's face it: a lot of the free things in CGI-land are complete programs.

Though I'm careful to target my audience to relatively low-end sites, my tirades are usually colored with particular emphasis on the portability of scripts in this new age of parity among Web server platforms. There's a necessary assumption in my sermons, though – you have to write them to be portable to begin with.

Perhaps the biggest culprit is sending mail out after processing a form. This has to be one of the most common tasks performed on Web sites. If you wander through the various script archives, you'll find an impressive collection of tools to do this very thing. Most operate quite simply: you create a form; then you create a template of what you want the mail to look like, often contained within the processing script itself. When you process the form, you substitute the form inputs in some manner on your template and shoot it off through the mail. In the UNIX world, it's routine to find scripts opening a pipe to the sendmail program. On Windows NT-based Web servers it may not be so easy, but several solutions have emerged in the past couple of years. I personally use a socket subroutine in my Perl CGI scripts that works in both worlds to satisfy my desire for the "Holy Grail of Portability."

Sometimes a solution to a common problem just boggles my mind with its pure simplicity and elegance. I've written dozens of variations of this form to mail things in Perl. They all work. In fact, they all work on any platform that runs Perl and has access to a mail server that's willing to process outbound mail. But while I periodically come up with some clever lines of code, the bottom line is that these things are rather verbose and downright ugly.

I recently had occasion to spend some of the wee hours after midnight on the phone with a colleague discussing how one might go about using ColdFusion to run one of these form to mail gizmos. What about sending back a filled-in version of the original form? If this were possible, it would have some interesting possibilities for a complete application. The task at hand was simpler, though. Let's have a look at how the two methods are similar and how they differ.

We'll begin by constructing a simple, fairly harmless form. We'll add a bit of mild complexity with checkboxes and radio buttons to show off some interesting syntax in the CFML version.

Our form is a brief hypothetical survey to be taken by coffee drinkers (see Figure 1). The source HTML can be found in Listing 1.

Figure 1: The survey form

The significant items in any form submission are the action and method attributes of the form tag and the name and types of the form field elements. In this case, the identical form can be used for either the CGI or CFML implementation simply by changing the action attribute of the form tag. In the CGI example, I've called the processing script "form_mailer.cgi" (see Listing 2); we'll switch the extension to ".cfm", as found in Listing 3, for a somewhat more elegant illustration. This said, we may use the same form simply by switching:

```
<form action="/cgi-bin/form_mailer.cgi"
method="POST">
```

to:

```
<form action="form_mailer.cfm"
method="POST">
```

After users complete the form, they click some sort of submit button to send the data to a script processor. The data is sent in URL-encoded format, which at its simplest means that we converted some nonprinting characters to their hexadecimal equivalent. On the processing side, we must reverse the encoding process to give us back the human-readable format and then feed that passed data into some sort of output format. This often takes the form of simply thanking the user and echoing back the form inputs for display purposes and taking that same echoed input and piping it to a system mail program. As mentioned earlier, this method is standard fare on UNIX, where a system mail-handling program is part of the operating system. It does raise some issues, however, when you desire to run that same script on the Win32 platform.

I take a somewhat different approach in the CGI method in that I favor using sockets to do the mailing. This ensures that my script is able to perform the mailing functions on any platform. A rough example of a Perl version of this processing script can be found in Listing 2. Readers will note that the traditional "shebang" looks a bit odd.

Instead of the typical "#!/usr/bin/perl", there is instead "#!perl.exe". I've gone the route of Apache on Windows NT as well as UNIX. Modules are available from Allaire to use the ColdFusion Application Server with

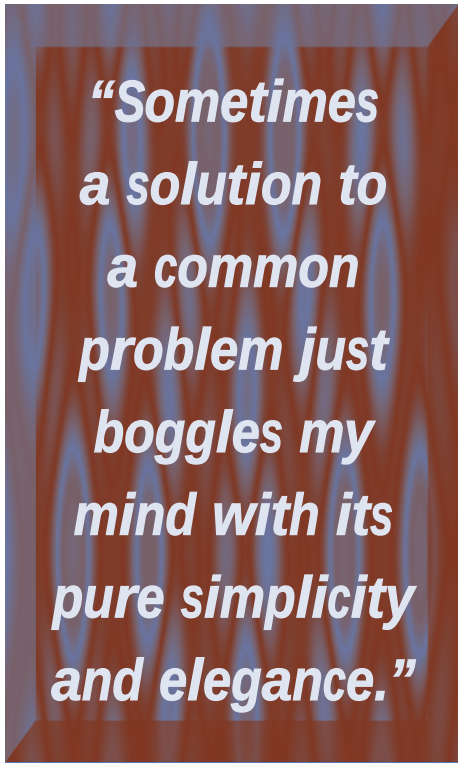
the Apache 1.3.x Web servers. Though the Apache Group is still (at time of writing), labeling this release of the Win32 Apache as beta code, I've been using it happily for many months.

Of particular note in the Perl script are the two rather ominous-looking subroutines dedicated to handling the decoding of the form data and the actual mailing. While they (or minor derivations thereof) are widely used, they do require a reasonably in-depth understanding of the language if you want the warm, fuzzy feeling of being in control of your code. One additional note regarding the socket mailing routine is also in order. An HTML content header has been added to the mail headers to facilitate the HTML-formatted mailing. For plain text mail, this should be removed and an additional carriage return added to the Subject header entry.

Let's move on to the ColdFusion version. We'll stay with the same form except for the change to the action attribute noted above. A quick read through the listings reveals two very distinct differences from the CGI version. First, the code to do the mailing can actually be read by the average human being. Second, there's no routine to decode the form data. In fact, if you've tinkered at all with using ColdFusion templates to process forms, you already realize that you get that part for free!

At this point, the two methods should feel more similar than different. Except for the built-in support for the moderately complicated decoding and mailing, the concepts are pretty much the same. What does add a bit of a twist are the form elements other than text inputs. These days, it almost goes without saying that we're always trying to capture data for use later on. In the early days of CGI, this meant simply sending the information to whoever was collecting it via the e-mail message, and what happened beyond that point was additional work.

This data capture to a database is where the ColdFusion version really starts to shine. In my earlier article on making the transition (***CFDJ*** Vol. 1, Issue 1), I illustrated using the ODBC.pm module to interface with a database. The syntax is admittedly a bit tricky, particularly to those not familiar with the object syntax of Perl modules, and you have to write out complete SQL statements. Additions to a database using ColdFusion can handle every field on the form with a single <CFINSERT> tag. This isn't so much a problem on a simple form, but consider some of the paper forms you've filled out in your life. If you've applied for any sort of financing or have purchased property, the forms involved can be overwhelming. Consider processing these types of



forms. Aside from the sheer size of these things, they're often riddled with boxes to check or leave blank, options to choose from and many yes/no questions. We translate these on the screen to checkboxes and groups or radio buttons. Handling these requires some thought up front to model the data correctly, but they're surprisingly easy to handle in the ColdFusion code.

My example assumes the receiver is using a mail client that is able to read HTML-formatted e-mail, and I simplified my life a bit by reusing the original form, just adding value tags to the form elements in the processing script or template to "fill out" the form. In the Perl example I'm forced to break up my output to insert some conditional code in order to handle the nontext fields for every radio button and checkbox. This generally creates large blocks of a conditional that are repeated, and change the action to be taken only if the condition is met. The logic works as follows:

```
if (this radio button checked on form) {
 create checked radio button
 create unchecked radio button
 } else {
 create unchecked radio button
 create checked radio button
 }
}
```

I've grown very fond of the equivalent CFML logic, which might read:

```
create radio button <cfif some test> check
the button </cfif>
create other radio button <cfif some test>
check the button> </cfif>
```

Note how, in the ColdFusion version, the conditional is embedded directly within the form tag itself. Where this becomes extremely helpful is when you're writing code that someone else will use. When writing out long blocks of HTML from a Perl script, I use what's called a "here document." The format looks something like this:

```
print << END_OF_HTML;
```

Put whatever you want here – variables get interpolated but everything you put here gets printed out.

```
END_OF_HTML
```

This works great because the nonprogrammer who wants to edit the HTML output doesn't need to know anything about the language as long as edits are confined to this block. It falls apart when we must create these over and over for very short bits of HTML, while needing to switch back to straight Perl statements to do the conditional tests. This document is a wonderful mechanism for writing long blocks of output, but unfortunately it will write out code snippets as plain text. I'll take elegant and functional over ugly and functional whenever I can get it!

Though the concept of mailing form input is a straightforward one, it's nonetheless one of the most frequent tasks performed on Web sites. Extending the form mailer to include archiving inputs to a database offers an extension to this task whose applications are virtually endless. I've recently taken this idea and extended it to an extensive forms application that allows geographically dispersed field personnel to maintain forms-based information nearly in real time without redundant writing of data by hand and expensive faxing of forms to a home office. From little seeds some great ideas grow. CFDJ

CODE LISTING

The complete code listing for this article can be located at **www.ColdFusionJournal.com**

About the Author

Jim Esten is the lead developer at WebDynamic, specializing in CGI programming, installation and maintenance. He also heads ColdFusion and Perl development for StarQuest Internet Services, a southern California Web hosting and dial-up services provider. You can contact Jim at jesten@wdynamic.com.

jesten@wdynamic.com

ColdFusion Develop

www.ColdFusi

er's Journal Forum

onJournal.com

CFDJnews

Able Solutions Announces AbleManager

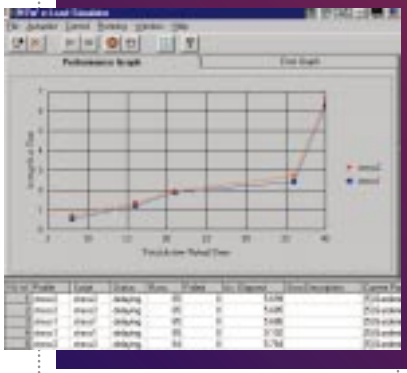
(Vancouver, WA) – Able Solutions, makers of AbleCommerce, has announced the release of AbleManager.

Built on Allaire's ColdFusion, AbleManager is a browser-based application that offers businesses direct Internet e-mail support. It's a contact management tool that offers flexibility and an easy-to-use interface that supports most popular browsers including Microsoft Internet Explorer and Netscape Navigator, as well as products like ProtoNet's ProtoFax.

For more information go to Able Solution's Web site at www.ablecommerce.com. CFDJ

Financial Services Leader Selects RSW Software

(Boston, MA) – RSW Software, a business unit of Teradyne, Inc., has announced that Countrywide Home Loans, Inc., has selected RSW's e-Test Suite to test the development and deployment of their Web application infrastructure.

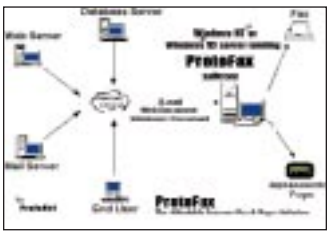


Under the terms of the agreement, RSW will supply its e-Test Suite to Countrywide for testing its critical online applications – a huge endeavor, as Countrywide has projected the launching of as many as a thousand new applications over the next two years. RSW's e-Test Suite will enable Countrywide to test its applications while still in the development stages, and after they go live.

For more information about RSW's e-Test Suite, check their Web site at www.rswsoftware.com. For more information about Countrywide go to www.countrywide.com. CFDJ

ProtoNet Develops Custom Tag for ColdFusion

(Woodland Hills, CA) – ProtoNet has become a Technology Partner with Allaire Corporation, and has developed a free custom tag for Allaire's ColdFusion.



The custom tag, called <CF_FAX>, closely integrates ColdFusion with ProtoFax, ProtoNet's flagship fax and pager product. Together, ColdFusion and ProtoFax allow ColdFusion developers to deliver any information handled by ColdFusion to fax machines and alphanumeric pagers.

ProtoFax is a software server package that turns a standard PC with a modem into an affordable Internet fax and pager server. Free copies of <CF_FAX> tag 1.1 and a free evaluation version of ProtoFax 3.1 can be downloaded directly from ProtoNet's Web site at www.protonet.com. CFDJ

Intermedia.NET Clients Build Online Business with ColdFusion 4.0

(Palo Alto, CA) – Intermedia.NET, a Web hosting provider supporting ColdFusion 4.0 and instant custom tag registration, announces two new client sites, www.CGI-Java.com and www.FINDaHOST.com. CDR Interactive Media developed the sites entirely in ColdFusion 4.0.

At www.CGI-Java.com, Web developers can quickly find CGI, Java or JavaScript resources, articles, tutorials and job opportunities. CDR Interactive Media developed the site entirely in ColdFusion 4.0 with an Access database.

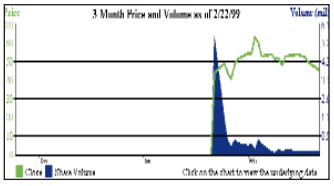
FINDaHOST.com is a directory of Web hosting companies. A visitor can check off items desired in a hosting plan, such as ColdFusion, ASP, SSL, etc., and determine the order of results to be displayed. Hosting companies control their listings; they log in, add information and adjust current listings.

To learn more about Intermedia.NET's services, visit www.intermedia.net. CFDJ

Allaire Corporation Reports Financial Results

(Cambridge, MA) – Allaire Corporation (Nasdaq: ALLR) has reported the results of their recent stock offering.

At the beginning of 1999, the company sold 2,875,000 shares of its common stock through an initial public offering. Net proceeds from the offering were approximately $52.5 million after deducting the underwriting discount and estimated offering expenses. The company intends to use the net proceeds of the offering for general corporate purposes.



For more information you can go to Allaire's Web site at www.allaire.com . CFDJ

Synergy 1.5 Delivers Truly Global Access to Information

(Vancouver, BC) – Catouzer announced the latest release of Synergy 1.5, its Web application framework. It features Synergy Anywhere, a simplified version of Synergy that allows users with older Web browsers to communicate in the Synergy collaborative work environment.

Large improvements have been made in terms of customizability of core collaborative apps. Synergy Anywhere was created by producing an object-based programming environment. This was accomplished by creating over 20 components that handle all database interactions and display/form generation tasks. Anywhere components also feature complete integration into the ColdFusion Studio 4.0 environment.

For more information on the many features and benefits of Synergy, visit their Web site at www.catouzer.com. CFDJ

CFUG	Contact Name	E-mail Address
Arizona	Christopher Kracht	cfug@aznet.com
Australia (Melbourne)	Marc Dimmick	marc@tved.net.au
Australia (Sydney)	Brent Pearson	b.pearson@morganbanks.com.au
California (Bay Area)	Nathan Dintenfass Kelly Ross	nathan@changemedia.com kelly@changemedia.com
California (Los Angeles)	Bruce Braunstein	bruceb@yahoo.com
California (Sacramento)	Allen Kinnard	akinnard@cwnet.com
California (Southern)	Leon Chalnick	lchalnick@prprpr.com
Canada (Vancouver, BC)	Cameron Siguenza	cameron@catouzer.com
Colorado (Northern)	Robi Sen	robi@granularity.com
Europe (Central)	Sven Slazenger	slazenger@interlake.com
Florida	Dale Mulert	dale@eng-tech.com
Georgia (Atlanta)	Cameron Childress	cameronc@mcrae.com
Illinois (Chicago)	Bob Burns	bob-burns@mediaone.net
Maryland	Michael Smith	Michael@teratech.com
Massachusetts (Boston)	Steve Casco	steve@ateaze.com
Minnesota (Twin Cities)	Dan Chick	chickd@fastcart.com
New Jersey	Cindy Pomarlen	pomarlec@sharpsec.com
New York (NYC)	Michael Dinowitz	mdinowit@houseoffusion.com
New York (Park Slope)	Bob Siegel	chairman@cfsig.org
North Carolina	Jon Owen	jon@webserve.com
Ohio	Angelo McComis	Angelo@CompuServe.com
Singapore	Chou Hao Chan	chiouhao@intracomm.com
Texas (Austin)	Dave Williams Miki Hardisty	david.williams@ci.austin.tx.us miki.hardisty@the401k.com
Texas (Dallas/Fort Worth)	Patrick Steil	pmsteil@imailbox.com
Texas (Houston)	Andy Dubinsky	andy@netfast.net
United Kingdom	Paul Underwood Mark Blair	paul@cfug.com
Virginia (Charlottesville)	Steve Nelson	m@secretagents.com
Washington, DC	Steve Drucker	sdrucker@figleaf.com

UK ColdFusion User Group

CFUG Spotlight

The UK ColdFusion User Group was founded over a year ago to provide a central resource of information for the CF community in the UK. Having been involved with the successful Visual Basic and Borland User Groups, I understand the issues surrounding the building of a good user group. User groups in the UK are very popular due to the problems involved with getting information from U.S.-based companies, and we have a number of people who reply as the voice of the user group as a whole.

Unlike the more formal user groups, we don't charge for membership. We try to meet at least every three months, preferably with one of the Allaire team to give a lecture on a particular topic. We also try to meet in smaller groups for a drink and to talk about various CF issues of the day.

The group is working on a project to build a central resource of information at www.cfug.co.uk. We have the www.cfug.com domain and hope at some point to link all the user groups from around the world there.

If you're interested in joining the group, please e-mail Paul Underwood at paul@cfug.co.uk.

by **Paul Underwood**

GET YOUR OWN!

Subscribe Today and receive the "CFDJ Digital Edition" FREE at www.COLDFUSIONJOURNAL.com

two years $89.99 12 issues

save $18

one year $49.99 6 issues

don't miss a single issue!

$69 one year Canada/Mexico
$79 one year all other countries

1 800-513-7111

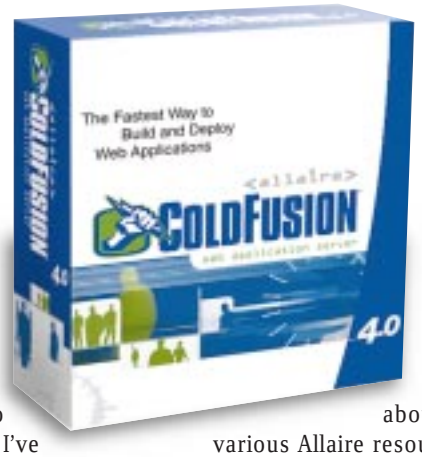
or subscribe online for faster service
subscribe@sys-con.com

ColdFusion 4.0
More on Developer Enhancements

The best just keeps getting better

by **Charles Arehart**

In the premier issue of ***CFDJ***, Richard Schulze wrote a fine article on several new features of ColdFusion 4.0, part of Allaire's recent upgrade to their entire product line. In this article I'll highlight some additional features of the new release. I'll also expand on his coverage of language features and Studio enhancements. And for those of you who missed Richard's article, I've included another brief review of the key new features.



A Major Product Upgrade

While 4.0 was released in October, some customers may only now be fully investigating its features. This is a substantial update, bringing significant new features that place CF at the vanguard of Web application development environments. With much-needed improvements in scalability, security, integration and rapid development capabilities, 4.0 solves several pressing problems for line-of-business CF implementations.

Still, with CF being used for ever more complex Web applications, it's reasonable for firms to exercise some caution in evaluating and implementing 4.0 and its many new features. As with any endeavor, with greater power comes a need for greater responsibility. While enabling a more capable, scalable and secure production application environment, 4.0 calls for consideration of how best to implement and take advantage of the new features. And to be honest, there are some significant changes that can affect existing applications if the new release is implemented too casually. Still, the new features are compelling and you'll want to get started with them as soon as you can.

The New Features

You can learn more about the new release in various Allaire resources (listed at the end of this article), as well as Schulze's article, mentioned above, but here are the most significant new features:

- **Rapid development**
  - *Team development system*
  - *One-step deployment*
  - *Interactive debugging*
  - *Two-way visiual programming*
- **Open Integration**
  - *Native database drivers*
  - *CORBA components*
  - *XML integration (WDDX)*
  - *Visual tools extensions*
- **Scalable deployment**
  - *Auto-failover*
  - *Dynamic load balancing*
  - *Open state repository*
- **Complete security**
  - *Integrated authentication*
  - *Access control*
  - *Server sandbox security*

I'll touch on some of these features in this article, but beyond the "big picture" features, there are many other topics that may interest CF developers as well.

Changes in Packaging and Marketing

The launch of 4.0 initiated several marketing and packaging changes. Note that the names of all the products have changed subtly, with the application server (sometimes referred to previously as CFAS) now being known simply as ColdFusion Server, and both the ColdFusion and HomeSite names have been compressed into single-word representations. There are new logos for all the products, a revamped Allaire Web site, substantial new marketing efforts and a comprehensive series of Road Shows and Regional programs.

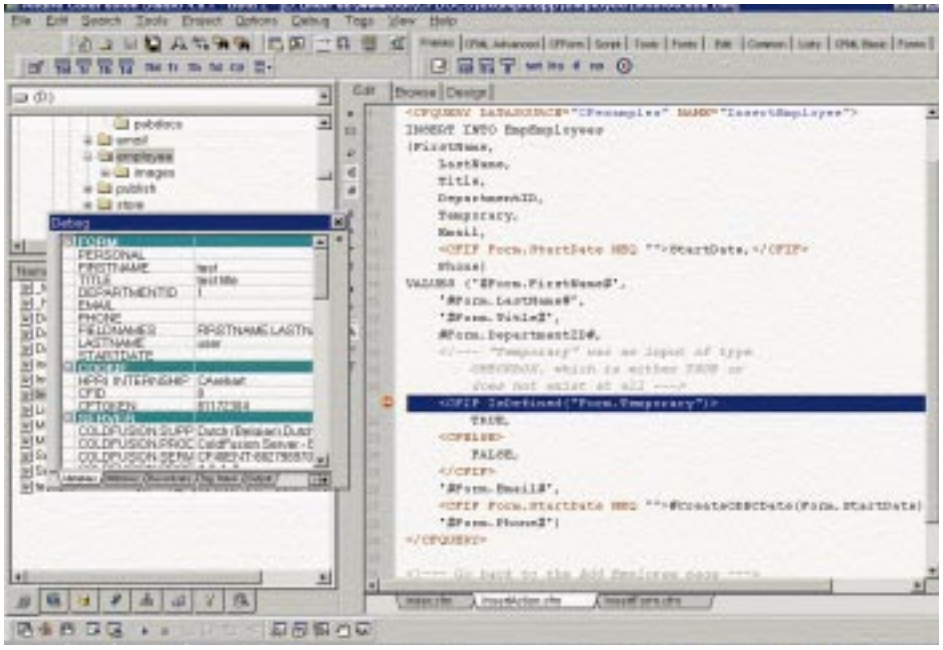
More significant to current clients is a new segregation of the ColdFusion product into Professional and Enterprise editions. Gone is the lower-end "workgroup" edition. The Enterprise edition, as its name implies, embodies those features most suitable to higher-end installations. Specifically, Enterprise includes features such as auto-failover, load balancing, native database drivers, CORBA components and advanced security.

Also changed is the price, befitting a more substantial new development system. It's been raised, though it's still a bargain compared to other similar Web development environments. (Frankly, many people got an awful lot of value out of the previous – and quite inexpensive – application server.) There were substantial discounts and generous upgrade paths offered at its launch, most of which have expired. Contact your Allaire sales rep for further details.

Studio Enhancements

While many CF developers have gotten by with simply coding CFML in text or generic HTML editors, it's time for them to reconsider that approach with the enhancements to ColdFusion Studio. Some of the new features mentioned above are in fact features of Studio, or features that benefit from tight integration of Studio with the CF server. Key among those are the new interactive debugging and team development capabilities and server sandbox security. There are many others as well.

The debugger is a fascinating new tool. Anyone who has developed even a modestly



CF Studio 4.0's new interactive debugger

complex CF application will have encountered situations in which it's been hard to tell just what was going on with an errant application – whether that was knowing just what code was being executed or knowing the value of a given variable, etc. The debugger solves both problems by enabling a programmer to set breakpoints (or single-step through code) and by monitoring the value of nearly any portion of the environment, including program variables, session and application variables and recordsets. This can also be valuable when you're reviewing someone else's complex code. The debugger can be a tad unstable at times, seeming to become stuck, but it's easily reset in such a situation by pressing the "end debug session" button and debugging again. The benefits well outweigh any first-release challenges.

The team deployment feature includes new integrated server-side source code control (SCC). Whereas previous releases of ColdFusion came bundled with Starbase versions – a fine LAN-based SCC tool – it became clear that Web-based team development was an entirely new animal that called for a more Web-centric, centralized and server-integrated approach. The new release incorporates server-side SCC via Microsoft Source Safe (not included) installed on the ColdFusion Server. (You can certainly continue to use existing client-side SCC packages.) Secure remote development via HTTP rounds out the package to provide more effective team deployment.

Finally, the Sandbox Security feature is an evolutionary – but substantial – change in that it's now possible to create a greater segregation of working environments among team members on the same server (or, more important, unrelated workers in different applications on the same server).

It might be easy to miss some other new enhancements to Studio because they're really enhancements to HomeSite and may be listed as such only in brochures and other descriptions (Studio is really just HomeSite on steroids, with added features to integrate with ColdFusion). Among the new features of HomeSite that "trickle up" to Studio users are:

- Tag inspector
- Tag tree
- Site view
- Codesweeper
- Bookmarking
- Style editor (CSS)
- Color coding of script code
- Support for SMIL and XML

The two-way visual programming aspect of Studio/HomeSite is interesting and well suited to quick prototyping and creation of initial layouts, but many CF developers will find the feature less than ideal for designing CFML templates. Design mode, as it's called, doesn't render your templates as they will look when CFML tags are executed, so most of the design benefits are lost. Additionally, code alignment can be unexpectedly modified if edited in design mode. There's a "codesweeper" feature that can be turned on to try to compensate for that, but it may be more trouble than it's worth for CFML templates. Try it for yourself.

A more welcome but perhaps easily missed change in 4.0 is that "projects" are no longer a means to create "virtual" collections of files from various directories. This approach had been problematic, especially in regard to keeping the project synchronized as files were added and removed. In 4.0, a project is simply any chosen directory (with optional subdirectories) that, once defined as a project, can then be quickly selected from the projects tab (eliminating the need to traverse the local or remote resource tabs). More important, projects have attendant features such as source code control and one-step upload capabilities (which include uploading only new or modified files as well as encrypting on upload). Projects can also be defined for remote servers (using RDS or FTP), which open such files to other project features, such as "extended find and replace" and "verify links."

Server Enhancements

Some of the new features in ColdFusion Server have clear benefits that are easily understood by most knowledgeable Web developers: automatic failover, dynamic load balancing, native database drivers, CORBA components, etc. The release notes are clear about these if you're not familiar with them, but others listed at the beginning of this article may still have you wondering:

1. *"Integrated authentication"* is an Enterprise capability that relieves developers of the burden of developing custom login features; more important, it can integrate with NT domains and LDAP directory servers or custom user databases.
2. *The "open state repository" capability* allows you to alter the location where the values of client variables (enabled via CFAPPLICATION CLIENTMANAGEMENT="yes") are stored on the server. Previously these were stored in the server's registry, but you may now indicate that they be stored in either a database table on the server or in the client's browser as cookies. This not only relieves the burden on server resources but also may open the door to using client variables in sites that previously restricted their use because of the registry burden, and it's the key to maintaining state in a clustered environment where a user's session may be passed across servers.
3. *XML integration* is primarily embodied in an interesting new feature known as Web Distributed Data Exchange, or WDDX. Whereas much of the industry seems stuck on the idea of XML as a replacement for HTML, Allaire has realized that XML's key role will be to facilitate the exchange of complex data between servers or to/from more intelligent client-side applications. WDDX is a clever, simple, yet effective new tool for just that purpose. With it, CFML variables, data structures and CFQUERY recordsets can be translated into XML (and vice versa)

to be passed to other servers, whether running CF or another server platform, or to a browser. It is powerful indeed. Also, a complete set of JavaScript functions is provided to facilitate the creation and use of these XML data structures in a browser-based application. See documentation on the CFWDDX tag for more info.

Several other new enhancements to the server don't warrant mention in some of the high-level, marketing-oriented literature that lists new features, but sometimes it's the little things that'll solve a key problem. Among these are:

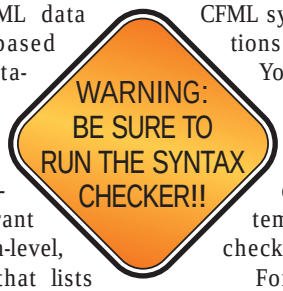
- *Improved exception handling*, which enables more effective and directed trapping and handling of errors
- *Associative arrays*, which enable the creation of complex but often easier-to-use data "structures"
- *CFCASE and CFSWITCH statements*
- *Greater support for DBMS stored procedures*
- *Support for OLE-DB for database connectivity*
- *Integration with the NT performance monitor*
- *Extended scripting*, which provides a means of programming within CFML that's not so tied to "tag-based" programming and the limitations caused by that approach

Schulze's article also described the powerful new concept of cached or "persistent" queries.

Issue to Note

Even with all the great new features of 4.0, there are still some issues that should be considered before the upgrade is implemented. First, be aware that (as of this writing) the Enterprise package's failover features don't yet work in an NT environment. This is a known issue and Allaire has pledged to offer a free 4.0.1 maintenance release soon to correct this. Similarly, the advanced security features couldn't be implemented initially in Solaris environments and will be corrected in 4.0.1.

Another new improvement to CF is support for stricter code validation. There are several reasons for Allaire to implement this, but the downside is that existing code may no longer validate in the stricter validation. Fortunately, they've offered two tools to assist programmers with this aspect. A new syntax checker can be used to quickly and easily test existing code without relying on execution errors to identify validation errors. You can choose whether to enforce strict validation on a server by way of a setting in the Administrator that allows you to turn it on or off – with a caveat.



Note: Even if you turn off Strict Validation in the Administrator, several changes in CFML syntax may cause legacy applications to fail if they're not corrected. You can't simply turn on 4.0 and turn off Strict Validation and expect all the applications to run. They likely won't. It's crucial that you run all existing templates through the new syntax checker to identify any such code.

Fortunately, you can easily direct the syntax checker (available on the page at start>programs>coldfusion server>welcome to coldfusion) to check all files in a directory (or those matching a pattern). The checker is installed by default in the server's CFDOCS directory and may also be executed directly, such as with http://yourdomain.com/cfdocs/cfmlsyntaxcheck.cfm.

Note: The checker reports only the first error in each template, so you'll want to rerun it after making changes until it reports no more errors in all checked templates.

It's imperative that you make this information clear to all developers on your server prior to implementing 4.0.

There are some other subtle changes in the new release. If you used CFEXIT to terminate a template's execution, you'll find that it doesn't terminate execution if it's executed within a template called by either CFINCLUDE or CFMODULE. The correct tag is CFABORT. (Technically, CFEXIT is for use in custom tags.) This behavior can cause real problems in code that depends on CFEXIT to terminate a process, such as in an included error-handling code fragment. Note also that CFEXIT has a new "method" attribute that can change its behavior still further.

Other Improvements

Still other changes fix limitations from previous releases. For instance, CFFILE now supports an "AddNewLine" attribute for the "Write" and "Append" actions. Set it to "No" to suppress adding a trailing newline character to the output. It defaults to "Yes" for backward compatibility. Those who tripped over this issue previously will be delighted by the news. And those who tried in previous releases to use CFPARAM to change a URL or FORM variable found that it wasn't allowed. In CF 4.0, these scopes are no longer read-only. This now makes it easy to indicate a default value for a URL or FORM variable if none is provided in the calling template. (The CGI scope remains read-only.)

A new CFSETTING option called SHOWDEBUGOUTPUT can control the display of debugging data within portions of a template when debugging has been turned on in the Administrator. And finally, a new variable called EXECUTIONTIME is available for CFQUERY. Used as QUERY.EXECUTIONTIME (not a "queryname." prefix), it returns the time it took for the last query to execute.

There are many other minor improvements like these. See the Release Notes, mentioned below, for more information.

Finding Out More

There are several ways you can learn more about CF 4.0. Of course, the documentation is always a good place to start in regard to technical details (and the docs in CF 4.0 have been improved substantially, with more examples, more detail and even new books). You can view these online at www.allaire.com/Documents/cf4docs.cfm.

In addition to ***CFDJ***, there are also Release Notes that explain several other issues of note at www.allaire.com/documents/cf4/allaire_support/releasenotes.cfm, as well as a "New Features" document that can be found at www.allaire.com/documents/cf4/allaire_support/cf4features.cfm.

If you stop at those, however, you'll be missing out. Another important document you should read is called the "Documentation Update." You can find it at www.allaire.com/documents/cf4/allaire_support/cf4doc_update.cfm. This not only lists corrections to the printed docs but, even more important, it lists some new features that weren't documented at all.

About 4.0.1

I'd previously mentioned some features that Allaire has publicly pledged to support in release 4.0.1, which is in beta as of this writing. Because of nondisclosure agreements, I can't release specifics regarding the update, but it's more than just a set of bug fixes. Be sure to check it out when it arrives. One thing I think I'm safe to speak of is that the hassle of not being able to test both the existence of a variable and its value in a single CFIF statement will soon be history. That makes 4.0.1 a must-have! Watch for it soon.

With 4.0, Allaire created a solid platform for the next generation of commerce-ready Web applications and has taken another step toward becoming the worldwide leader in platform-independent Web development tools and application servers. CFDJ

About the Author

Charles Arehart, a senior ColdFusion developer, trainer and researcher, recently joined Fig Leaf Software in Washington, DC. He's worked on enterprise database applications development and administration for nearly 16 years and has written and spoken for nearly as long on getting the most from the tools he's worked with.

carehart@figleaf.com

Highlight your website with
infoboard
NT and UNIX
Cold Fusion Hosting
Development Consulting
Oracle, Informix, MS-SQL, E-Commerce Plug-ins
<allaire> Alliance Partner
1-800-514-2297
sales@infoboard.com
www.infoboard.com

An ad in the ColdFusion Marketplace can bring you more business! Expose your product or service to over 30,000 registered ColdFusion users.

For more information, contact Robyn Forma at 914 735-0300 or robyn@sys-con.com.

Able Solutions

Enter the realm of browsable store building and administration – from your browser. Build "your_site.com" with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA
***www.ablecommerce.com** • 360 253-4142*

Allaire Corporation

At Allaire, our focus is to empower developers with the tools and knowledge to deliver on the promise of the Web as a platform for crucial business applications. We offer a wide range of flexible programs, including professional education, consulting, technical support and partner programs.

One Alewife Center, Cambridge, MA 02140
***www.allaire.com** • 888 939-2545*

Catouzer, Inc.

With the Synergy 1.0 Web application framework, creating custom Intranet applications is a breeze. The Synergy Application Development Kit (ADK) gives you the tools to rapidly develop your custom applications, which can be fully integrated and managed under the Application Services Layer (ASL).

1228 Hamilton Street, Suite 501, Vancouver, B.C. V6B 2S8, Canada
***www.catouzer.com** • 604 662-7551*

Eprise Corporation

If your customers are looking for a content management solution, Eprise Participant Server FastStart Kit for Allaire ColdFusion Developers can save you time and resources. Participant server is a flexible content management framework that enhances high-value business relationships through the delivery of timely, targeted, Web-based communications.

1671 Worcester Road, Framingham, MA 01701
***www.eprise.com** • 800 274-2814*

Fusion FX, Inc.

Fusion FX, Inc., is a full service Web site design and hosting company. Our strength in ColdFusion Development, as well as with custom AbleCommerce Store building, and the fact that we have a Team Allaire member on staff will give you peace of mind because you know your ColdFusion development is in good hands.

819 Peacock Plaza, Suite 535, Key West, FL 33040
***www.fusionfx.com** • 800 780-2422*

The Igneous Group, Inc.

The Igneous Group provides custom solutions for e-commerce business information systems and dynamic content publishing. We build systems to your specifications and integrate them with your existing backoffice data and business infrastructure. We can help you get your site online, or identify how to make it function more efficiently.

541 Seabright Avenue, Santa Cruz, CA 95062
***www.igneous.com** • 877 469-ROCK*

Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303
***www.intermedia.net** • 650 424-9935*

Live Software

The power of Java and the simplicity of ColdFusion, <CF_Anywhere> gives ColdFusion programmers the ability to leverage all the power and flexibility of Java using the familiar ColdFusion Markup Language (CFML). <CF_Anywhere> was perfected for CF developers, CF administrators, ISPs and everyone.

5703 Oberlin Drive, Suite 208, San Diego, CA 92121
***www.cfanywhere.com** • 408 996-0300*

Netamorphasis Corporation

Netamorphasis Corporation delivers and services software applications for companies who recognize the value of information technology. We specialize in the construction of custom database applications, intranet/extranet applications and the integration of legacy and Internet systems.

1270 Broadway, Suite 1108, New York, N.Y. 10001
***www.netamorphasis.com** • 212 279-6409*

ProtoNet

ProtoNet is the developer of ProtoFax, an Internet fax and pager server. ProtoFax allows companies to automate incoming orders from the Internet, and is perfect for e-commerce applications. Additional features include an intelligent fax queue and improved integration with proxy servers.

22900 Ventura Boulevard, Suite 175, Woodland Hills, CA 91364
***www.protonet.com** • 800 551-0636*

RSW Software

RSW Software is a wholly owned subsidiary of Teradyne, Inc., and specializes in Web application testing software. Established with the goal of providing best-in-class testing products, RSW offers a suite of products called the e-TEST Suite, which automates the process of testing business-critical Internet and intranet applications.

44 Spring Street, Second Floor, Watertown, MA 02172
***www.rswsoftware.com** • 508 435-8000*

Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003
***www.virtualscape.com** • 212 460-8406*

COLDFUSION MARKETPLACE

NetObjects Fusion 4.0

from NetObjects Inc.

Designing great-looking Web pages couldn't be easier

by **Shawn Cowman**

As I sat down with NetObjects Fusion 4.0 for the first time, I was curious to discover what it could offer a professional Web designer and host like myself. What I found was an impressive product for creating sophisticated business Web sites targeting electronic commerce.

NetObjects Fusion 4.0, which has been touted as "The fastest, easiest way to build business Web sites," really measures up. It enables you to build, manage and update Web sites immediately and efficiently.

Time is of the essence to Web designers. With increasing awareness as to what the Internet is and its incredible potential, more and more businesses are clamoring to go online. NetObjects Fusion's WYSIWYG environment lets you build a business site quickly. It automatically generates HTML and script code, which is required by a browser to display pages or to connect to databases and applications. The drag-and-drop SiteStructure Editor makes it easy for anyone to use. Rearranging, adding or deleting pages can be accomplished without the worry of broken links.

One of the toughest tasks faced when creating a Web site is producing a dynamic and interactive site that will peak the user's interest. NetObjects Fusion 4.0 addresses this with its smooth integration of third-party products and technologies within the new Java-based component architecture. The connectors that are bundled with 4.0 are iCat's Commerce, IBM HotMedia, Lotus Domino, Microsoft Active Server Pages and Allaire's ColdFusion. These connectors make it easy for developers to build an impressive site that includes e-commerce applications for ordering and payments. As a big advocate of Allaire's ColdFusion Application Server, I was especially thrilled to find this connector offering Web site dynamics without my having to spend hours writing complicated scripts. The same applies to the ASP connector, which forgoes the tedious script writing for Java-based wizards.

Figure 1: The SiteStructure Editor

In reviewing Fusion 4.0, I built a test site in order to play around with it. The results were a very professional-looking business Web site created in a little over four hours of design time. I especially appreciated the new table tool, which offers HTML table specification while supporting my desire to customize the HTML for cells, rows and columns.

Another feature, which I was really pleased with, was the ability to drag and drop a database right onto the page layout for live connectivity. I also had full control over the database table by using the Properties palette. The ease of use coupled with user control, which NetObjects Fusion 4.0 renders, makes it a definite asset to professional Web designers.

NetObjects Fusion's five main functions are Site, Page, Style, Assets and Publish.

The SiteStructure Editor displays a tree structure and outline of your complete Web site (see Figure 1).

Dragging and dropping page icons allows you to add pages, insert templates or import an entire site or any portion of it to NetObjects Fusion.

The Page View feature lets you lay out pages in whatever mode you choose. For text pages, the text-based editor serves as a word processor. If hand-coding is required, HomeSite 4.0 is included for HTML integration. If a page is graphics-intense, the graphics editor is available with rules, guides, alignment and an intuitive interface. Graphics can be placed precisely where they'll appear on the page with pixel-level control.

The SiteStyle Editor offers programmers hundreds of professional-looking site templates. SiteStyle elements include Banners, Primary Navigation Buttons, Secondary Navigation Buttons, Mouseover Buttons, Backgrounds, Lines, DataList Icons, Text Color, Link Text Color and Visited Text Color. You can also customize a design or create your own.

The Assets Manager contains your image file indexes, which allow you to cen-

Figure 2: A Data Publishing popup window

Figure 3: NetObject Fusion 4.0's banner builder

trally manage the resources in your Web site.

The Publish feature permits you to upload your full site or just updated pages to any server with Everywhere HTML (see Figure 2). NetObjects Fusion generates HTML that can be optimized for browser versions 2, 3 or 4. For sites using DHTML, Fusion 4.0 creates code that works with both Microsoft and Netscape 4 browsers.

To make your Web site more profitable, NetObjects Fusion components enable you to build a site that works with turnkey e-commerce systems such as Breakthrough. New to 4.0 are e-commerce elements that drag and drop onto the Layout Editor. You can now link an iCat Commerce Online Server to your site or use the included IBM Media to create a powerful online shopping experience. Also available are forms for collecting customer information, which are constructed using a special toolbar for dragging and dropping buttons, text-entry fields and other components. Fusion 4.0 also offers a JavaBeans tool that gives you full control over its actions, and an ad banner tool for creating eye-catching rotating banners (see Figure 3).

With the release of Fusion 4.0, NetObjects launched a Web site to assist designers in utilizing the full potential of the program. EFuse.com offers Web site building tips, tricks, articles, discussion groups and much more. EFuse.com is a great forum for any designer building a business site.

NetObjects Fusion 4.0 is an invaluable tool for both novice and accomplished Web site designers alike. Its ease of use makes it possible for anyone to create a professional business site with the potential of realizing immense Internet success. As a Web designer and host, NetObjects Fusion's open site environment allows me to maintain the creativity and flexibility of site design that I want, while integrating the latest technologies to create compelling, dynamic Web sites. I look forward to utilizing NetObjects Fusion 4.0 in my pursuit of achieving an optimum Web presence for my customers. CFDJ

About the Author

Shawn Cowman is co-owner and senior Web developer of intelistor, a Web hosting and design company specializing in ColdFusion development. Shawn can be reached via e-mail at scowman@intelistor.com.

scowman@interlistor.com

ProtoNet

www.protonet.com

Igneous Group

www.igneous.com

New Business Models on the Web

How to get there from here

by **Jeremy Allaire**

Growth in Internet commerce continues to fuel an adoption frenzy of Web technologies. The benefits of moving a business to the Web seem clear, but only a few can truly claim that they're building sustainable, long-term businesses.

The most prevalent form of money making on the Web seems to be online advertising. Any company with a site and an audience is selling online eyeballs, mostly through the online advertising networks. Even small sites with little traffic rely on online ads with click-bartering networks such as the LinkExchange.

But the novelty is wearing off. Even with the growth of Internet traffic and e-commerce, online advertisers seem to be reaching an upper limit in paying for eyeballs. Many attribute this to consumers getting tired of banner ads, or click-through rates being on the decline. To many skeptics, this trend is the swan song of online businesses.

On the Horizon

Just as this "decline" in ad-driven models is taking place we've started to see a variety of money-making instruments available to online businesses. These new models aren't just more of the same. Many, in fact, are driving new ideas in overall business models and opening up entire new economies.

Coined "Affiliate and Syndication Networks", these models (theoretically) enable any Web site to leverage the assets of other sites and create new value. An example of this was pioneered by Amazon.com, who years ago introduced an "affiliate" program. With this program, any site owner could become an online reseller of Amazon.com merchandise. The site owner simply created an affiliate account with Amazon, then pasted content and links on his site back to Amazon. If endusers clicked through this content and bought a book from Amazon, the site owner got about 20% of the revenue on the sale. It's Business 101: Share revenue and increase volume using a network of resellers.

But the idea expands when one realizes that this model could be at the core of a revolution in how we view the Web. Based on Metcalfe's Law (which states that the value of a network [the Internet] increases exponentially for every n-node added to it) this is the basis for a shift in the development of online businesses.

In this view, every site is a potential asset for your site. Instead of viewing the Web as islands of content linked only on the surface, this view sees the Web as an interconnected network of value, with subsurface-level links between every site. In short, it's a new economy built from the Web.

The Next Level

Taking your Web business to the next level requires Web syndication and affiliate networks. Getting there requires introspection on what aspects of your business support this model. For example, consumers and e-commerce providers have a clear model for exposing content and commerce transactions to site partners. Other examples might include exposing classic customer leads and contact info to exclusive partners via an extranet that uses an XML interface for sharing data. If your company has a technical product, perhaps the answer is providing an interface to your knowledge base and customer support databases for external companies who support your product.

And, once you've figured out what assets you want to exchange and expose, getting there is simple. As I discussed in my first ***CFDJ*** column, the Web Distributed Data Exchange (WDDX) provides a fantastic facility for exchanging data with other Web applications. If you aren't familiar with WDDX, go to www.WDDX.org and download the WDDX SDK (available for NT and UNIX).

Building an SDK

Once you've absorbed the basic mechanics of WDDX, you're on your way to building a custom SDK for your site. In this model, you provide your site affiliates with a custom Software Development Kit (SDK) for remotely accessing services and data from your site. Imagine this as the "Amazon SDK": a set of custom scripts for remotely integrating the Amazon.com back-end into your custom front-end site.

Developers will expect their partners to provide out-of-the-box libraries for accessing Web applications and data. Custom SDKs will become the norm for doing this.

The 10 Steps

1. *Learn to use the WDDX SDK.* This is key, as it provides a common way to expose data assets of your site to other sites.
2. *Investigate your own site.* Explore your application and determine the points of entry other sites might want to use.
   For example, in a commerce site, the browse and search interfaces for catalogs are a logical place to start. Syndication also works for the traditional order form.
3. *Create an "affiliate tracking system".* Simply, you need a way to generate an ID which a remote site can use to identify itself when remotely interacting with your site. This ID should be used to track affiliate activity. Using this data, it's easy to see who your successful affiliates are, track any revenue-sharing implications and build activity profiles.
4. *Add a WDDX API to your site.* In many cases this is as simple as adding conditional logic to your existing applications. For instance, in a catalog application, if the affiliate ID has been passed into the page on the URL, you'd know that it's a remote application requesting data. You'd then grab the incoming WDDX packet and Deserialize it into local CF variables. From this you could determine what products they're interested in and execute the relevant CFQUERY. Instead of returning an HTML page, you'd return a WDDX packet containing the Recordset for the remote application's consumption.
5. *Build script wrappers of your WDDX API.* You'll want your site's SDK to be easy to use. To do this, give your site partners simple components that are easily used in any environment. Since WDDX supports virtually every Web application environment, this should be easy. First pick your platforms. I'd suggest starting with Perl, ColdFusion and ASP. Technically, you'd build custom components in each environment, which in turn would wrap the WDDX interaction with your site. The components are your site's "API".
6. *Document your site's API.* Like any good product, your site's SDK needs to be usable, and the key to usability is user education. Document your APIs in an HTML format.
7. *Package it all up.* Your affiliates should simply grab a single package of scripts from your site, drop them on their servers and be off. To do this you'll need to package up the components of the WDDX SDK with your SDK. Your site partners shouldn't have to learn anything about WDDX or the WDDX SDK, but instead get your branded package of scripts. The good news is that the WDDX SDK and all its components are all freeware and open source code. You can take these and do whatever you want; even distribute them with your own commercial product!
8. *Let people know.* Once you've got an SDK running, put an icon on your site letting people know that if they want to be affiliate partners, you've got a whole program and SDK available for use.
9. *Let us know.* We're excited about all the WDDX projects happening out there, and we want to know about yours. Let us know at projects@wddx.org.
10. *Have fun reinventing the world.* Not only are we all in this to reinvent our businesses, but we've all got broader ambitions, no doubt!

About the Author

Jeremy Allaire is a cofounder and vice president of technology strategy at Allaire. He helps determine the company's future product direction and is responsible for establishing key strategic partnerships within the Internet industry. Jeremy has been a regular author and analyst on Internet technologies for the past seven years, and he holds degrees in both political science and philosophy from Macalester College.

jeremy@allaire.com

Catouzer

www.catouzer.com

Intermedia.net

www.intermedia.net